RICARDO STANICIC

175 PTAS

mi computer

78

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR



Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VII-Fascículo 78

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones Francisco Martín
Asesor técnico:	Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco

Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-067-2 (tomo 7)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 108507
Impreso en España-Printed in Spain-Julio 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



Viajar a otros mundos

Terry Kennett

La magia de aprender
Los juegos de aventuras y simulaciones de carácter educativo permiten que el niño explore, en el propio entorno de la clase, situaciones cuya observación en la vida real es inaccesible o potencialmente peligrosa. Lejos de ser una pérdida de tiempo, la experiencia sugiere que tales juegos pueden favorecer considerablemente el proceso del aprendizaje

Los juegos de simulación y de aventuras permiten al estudiante acceder a esferas que, de otra forma, serían inalcanzables para él

Los juegos son divertidos, y todos los niños y la mayoría de los maestros coinciden en que el aprendizaje también debería ser entretenido. La utilización de juegos en general y la representación de papeles en particular se han popularizado en los últimos años, paralelamente al auge que han experimentado las actividades teatrales en la educación y el desarrollo de la "teoría del juego" en psicología. Estos desarrollos le confieren aún más credibilidad al potencial del ordenador en la escuela.

El juego del tipo "marcianitos" y la más reciente variedad de los juegos de aventuras/simulación representan los dos géneros dominantes de los juegos por ordenador. A medida que se va desarrollando el software, las diferencias entre ambos van volviéndose cada vez más sutiles, pero la mayoría de los juegos todavía se pueden reconocer como inspiradores en uno u otro tipo de programas.

Aquellos pertenecientes a la variedad de conflictos espaciales tienen un impacto muy leve en el software educativo, a pesar de su popularidad. Al escribir su libro *Mind at play. The psychology of video games,* Elizabeth y Geoffrey Loftus interrogaron a numerosos psicólogos acerca de la efectividad educativa de esta clase de juegos: las respuestas que obtuvieron no fueron alentadoras. Como cabe esperar, el valor de los juegos de "marcianitos" como fuente de entretenimiento sobrepasa cualquier posible valor educativo, si bien la investigación demuestra que esta clase de juegos, debido a la coordinación y los reflejos que exige, puede tener efectos terapéuticos para quienes sufren de diversos tipos de enfermedades mentales y lesiones cerebrales.

Programas como *Spell invaders* (Los invasores de la ortografía) y *Maths invader* (El invasor de las matemáticas) han sido intentos de presentar temas tradicionales en formato de "invasores del espacio", teniendo el niño que "disparar" las respuestas correctas. Pero estos juegos han demostrado ser menos divertidos que los originales y su eficacia educativa es, cuanto menos, dudosa.

Cortesía de Simulation-Database Project, Glasgow College of Education

Ian McKinnell

Un viaje a través del tiempo
El Simulation-Database Project, desarrollado en el St. Andrew's College of Education de Glasgow, en Gran Bretaña, utiliza el ordenador como herramienta para complementar (y no sustituir) otras actividades del aprendizaje. El paquete se divide en tres partes: un "juego de aventuras", un conjunto de archivos de texto que proporcionan información adicional y una base de datos que contiene referencias a otras fuentes. La clase se divide en grupos, asumiendo cada grupo el papel de un personaje que (en este ejemplo) realiza un largo viaje a través de la Escocia del s. XIX. Las actividades adicionales que se derivan del uso de la base de datos, y que a la vez la acompañan, incluyen trabajo con mapas, escritura creativa, actividades teatrales y excursiones

Las simulaciones y las aventuras, por otra parte, demuestran un potencial considerable como alternativas educativas. Las simulaciones utilizan el ordenador para crear entornos de aprendizaje que de otra forma no estarían al alcance del niño, y los juegos de aventuras pueden reproducir los escenarios fantásticos de la ciencia-ficción y la mitología sin los gastos o los peligros inherentes a la realidad. Estos principios ya se han aplicado en los planes de entrenamiento industrial y en la actualidad muchos de ellos se están utilizando en las escuelas.

Ballooning (Aerostación), un programa de Hill McGibbon para el Spectrum y el Commodore 64, es un típico juego de simulación que le ofrece al niño la oportunidad de aprender jugando con algo a lo cual, de otra forma, jamás tendría acceso. En un panel de instrumentos situado en la parte inferior de la pantalla hay un indicador de combustible, un altímetro, un indicador de temperatura y un dial de velocidad de descenso/ascenso, que actúan de la forma adecuada cuando se enciende o se apaga el quemador, o cuando se abre y se cierra la válvula de gas. Cuando el globo aerostático desciende a una cierta altura se hace visible el terreno y, en el caso de que fuera necesario un reabastecimiento de combustible, hay varias estaciones de aterrizaje donde se podría hacer descender la nave.

Ventajas educativas

Además de ser económicas y seguras, las simulaciones ofrecen otras ventajas educativas. Representados en un ordenador, ya se han tenido en consideración los detalles de preparación de un experimento o ejercicio. Por consiguiente, los niños quedan libres para concentrarse en la situación, en lo que está sucediendo y por qué. El tiempo ya no es un obstáculo que impida completar en clase proyectos complicados y que consumen mucho tiempo. Experimentos científicos que pueden requerir días en el laboratorio se pueden observar en la pantalla en cuestión de segundos. Un modelo por ordenador simplifica los sistemas que por su complejidad resultan difíciles de comprender para los estudiantes, y los gráficos pueden realzar las funciones pertinentes con datos representativos de las etapas importantes del proceso. Dado que la simulación se puede repetir en circunstancias diversas, los estudiantes pueden experimentar de forma indefinida. La dirección en que avanza el experimento, y los factores variables implementados, al estar todos bajo el control del estudiante, permiten una mayor comprensión de los resultados y las conclusiones y, puesto que la simulación se presenta como un todo y no en partes, se obtiene una perspectiva más amplia.

Existen, no obstante, ciertas limitaciones inherentes a las simulaciones. Debido a que un ordenador puede almacenar sólo una cantidad predefinida de variables, existe el peligro de que una simulación presente una visión demasiado simplificada de su tema. Por consiguiente, la capacidad de describir situaciones de la vida real es al mismo tiempo su mayor cualidad y, tal vez, su mayor punto débil. En realidad, es lo imprevisto lo que nos sorprende y nos obliga a reevaluar nuestro conocimiento sobre lo que está sucediendo. Por este motivo, un programa como *Ballooning*, con su cantidad fija de variables, ofrecerá sus mayores ventajas cuando se lo utilice en un contexto más amplio, quizá como un único aspecto de la ciencia y la historia del vuelo. Lamentablemente, esta limitación con frecuencia se pasa por alto; una simulación no es un sustituto de la realidad. Su punto fuerte consiste en estimular a una mayor indagación de los temas relacionados, no en proporcionar respuestas ya elaboradas o como una simple ampliación de las técnicas de aprendizaje programado. Hay, por ejemplo, un



programa de ciencias escrito para el Apple, que simula un proyecto para la conexión de bombillas y pilas de diferentes maneras. Presumiblemente, se ha escrito para la escuela que pueda permitirse un ordenador caro pero que, por algún motivo misterioso, no tenga acceso ni a bombillas ni a pilas. El posible shock de una pila de 6 V apenas puede justificar la simulación por cuestiones de seguridad.

Creación de microcosmos

La mayoría de las personas, niños y adultos por igual, disfrutan recreándose en la fantasía. En los últimos años, libros como *El Hobbit, 2001: una odisea del espacio* y la serie *Conan*, de Robert E. Howard, han adquirido una notable popularidad. Juegos de representación de papeles como *Dungeons and dragons* (Calabozos y dragones), no sólo han obtenido un extraordinario éxito comercial, sino que también han coincidido con el masivo reconocimiento del valor de la representación de papeles en la terapia y la educación. Su naturaleza interactiva hace que el ordenador sea un medio ideal para los juegos de fantasía y actuación. Las capacidades de gráficos y sonido pueden definir de forma más cabal un mundo histórico o imaginario. Los especialistas en educación han sabido ver rápidamente este potencial del ordenador para crear microcosmos, considerándolos como un salto cualitativo en las prácticas de la enseñanza.

No parece, por ejemplo, haber forma mejor de que un niño aprenda historia que retrocediendo en el tiempo y viviendo una aventura. Pero el niño necesitará información para sobrevivir y, por tanto, el programa debe contener una base de datos que se pueda consultar durante el juego y utilizar como obra de referencia al margen del teclado. El Simulation-Database Project lo puso en marcha Allan Martin en el St. Andrew's College of Education de Glasgow, en Gran Bretaña, para "explorar la forma en que los ordenadores pueden contribuir significativamente al trabajo de la escuela primaria, mediante la creación de un nuevo tipo de objeto pedagógico basado en el ordenador: la base de datos de simulación". La estructura de la simulación está basada en un juego de aventuras: un niño entra en un mundo histórico en el cual, al viajar a través de diferentes zonas, ha de hacer frente a diversas situaciones e incidentes. Los archivos de texto de la base de datos están relacionados con cada una de las zonas y se puede acceder a ellos en cualquier momento. Los programas están diseñados para ser utilizados como parte de un proyecto mayor en el que la mayor parte del aprendizaje tendrá lugar fuera del ordenador, que es empleado sólo como uno entre muchos recursos.

Una segunda base de datos contiene referencias a la zona o el tema, y ésta puede ser consultada para obtener referencias a fuentes adicionales de información en libros, películas y otros medios de comunicación. Para ayudar a integrar el programa con otras actividades de la clase, el software incluye un libro de referencias, un mapa de la zona, cartas simuladas, imágenes y diverso material de referencia. Escritos en LOGO, los programas se pueden adaptar a las condiciones locales: es posible añadir zonas y datos, suprimirlos o alterarlos para adecuarlos al objetivo concreto de la escuela o clase en particular.

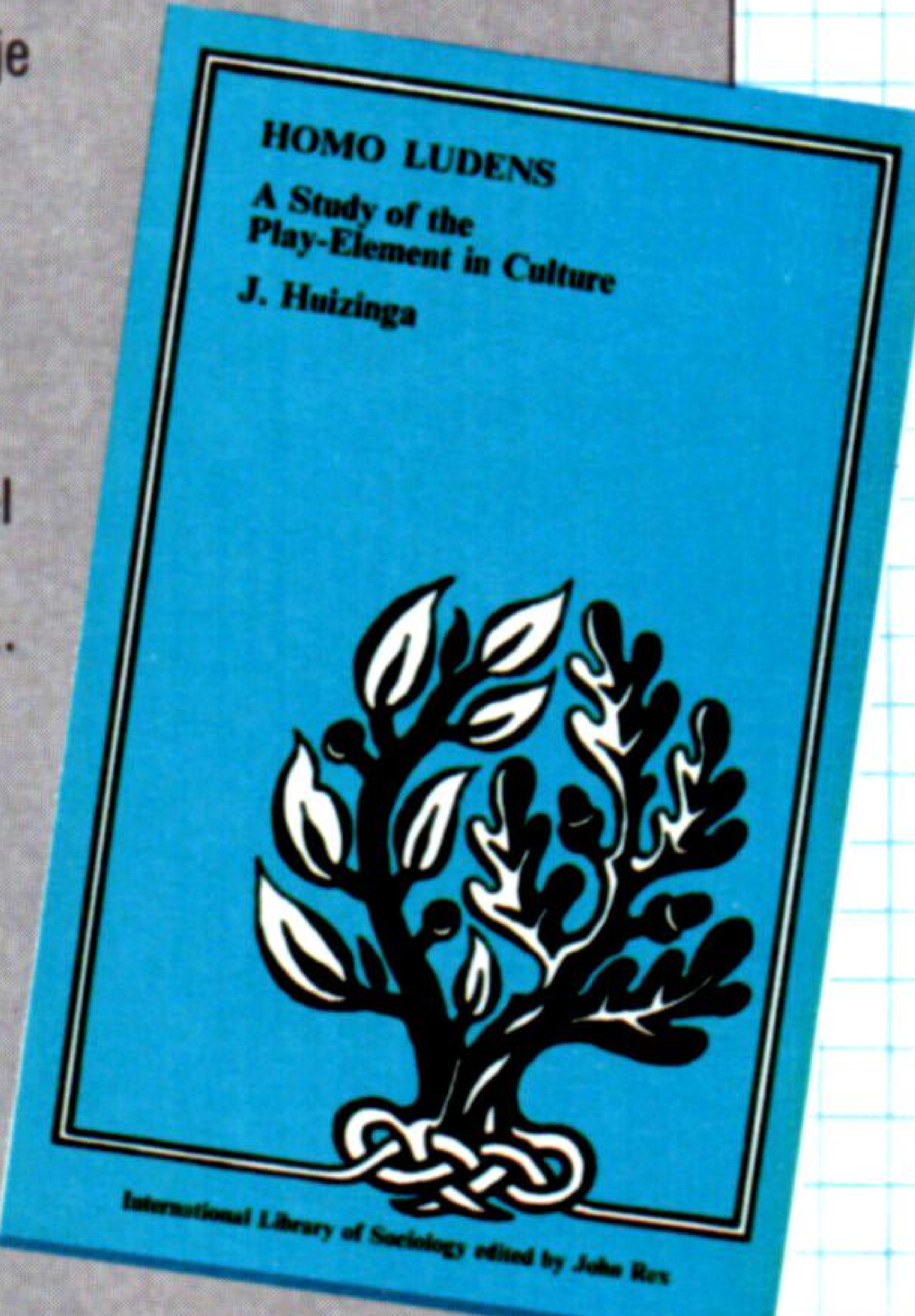
La primera base de datos simulada se basó en la Escocia de 1880. El paquete se utilizó en una clase dividida en seis grupos, asumiendo cada uno la identidad de un personaje ficticio pero inspirado en la realidad. Los personajes eran representativos de cada estrato de la sociedad, desde el aristócrata a la criada, y a cada grupo de alumnos se le dio un argumento que les exigía realizar un viaje a través de Escocia. Una vez que habían planificado su viaje con el mapa y se habían desplazado hasta la primera zona del ordenador, los personajes se tropezaron con diversos incidentes. Después de cada fase de movimiento, los niños, utilizando dos bases de datos y otras fuentes de referencias que les proporcionaba el maestro, investigaban la zona a la que habían llegado. Las actividades que generaban estas investigaciones incluían trabajo con mapas, redacción de diarios, redacción de informes de periódicos y otros trabajos descriptivos. Los niños producían obras, grababan "entrevistas" con los personajes, realizaban visitas a las áreas que cubría el juego e invitaban conferenciantes a la escuela. Favoreciéndose el debate durante todo el juego, así como el dibujo y la pintura de sus nuevos entornos, los estudiantes y el maestro disfrutaron con el desarrollo del ejercicio.

El proyecto de la base de datos de simulación constituye un ejemplo de cómo se puede utilizar un juego por ordenador para despertar en los niños el interés y estimular su trabajo en estudios sobre los temas relacionados.

El hombre y el juego

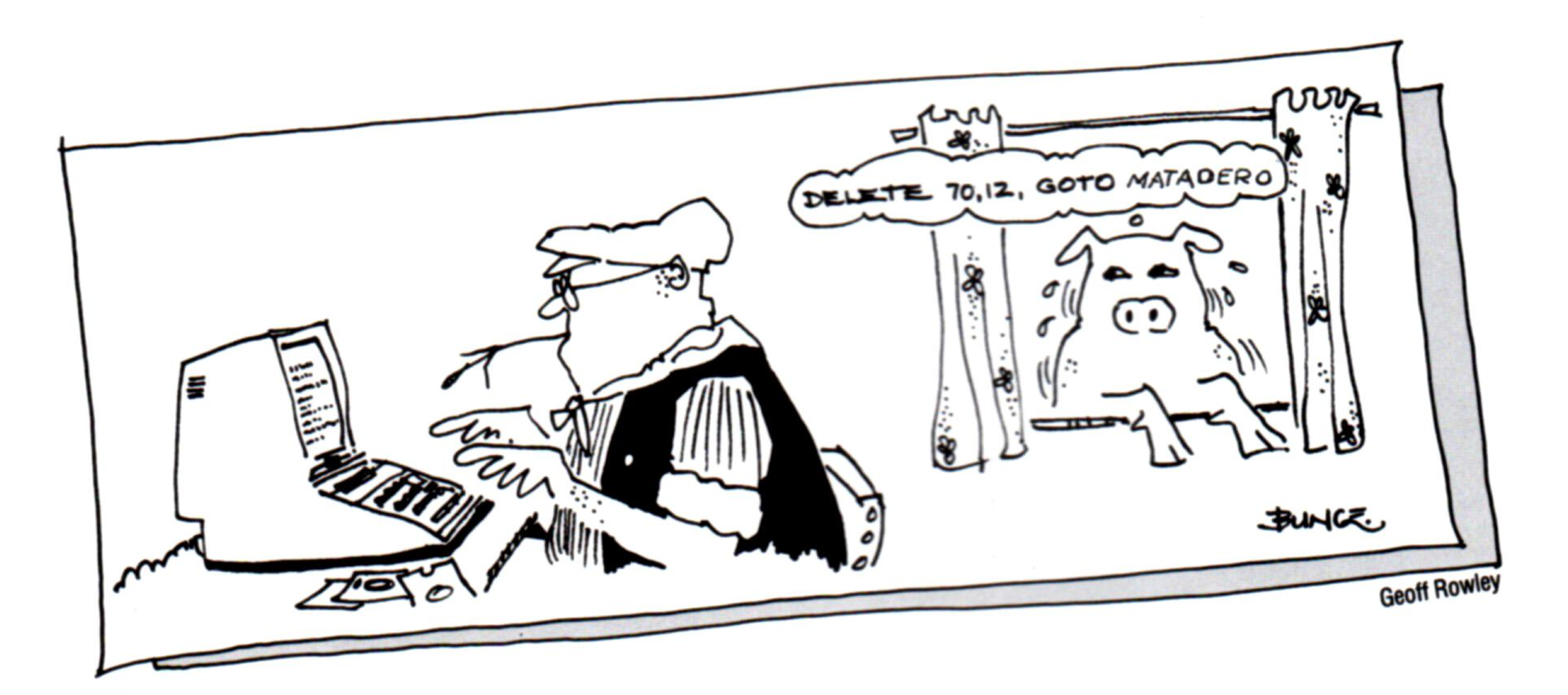
Muchos educadores y padres temen la posibilidad de que los estudiantes dediquen demasiado tiempo a "jugar" en lugar de "trabajar". Un reciente programa de televisión mostró a un grupo de niños explorando los gráficos de tortuga en el suelo de su clase. El comentarista advirtió a los telespectadores que no se debían dejar engañar por la naturaleza desestructurada del ejercicio al decir: "En realidad, aquí se está produciendo un verdadero aprendizaje."

No obstante, muchos padres y maestros están comenzando a creer que el "auténtico" aprendizaje sólo se produce a través del juego. Este punto de vista tan radical tiene una base clara, no sólo en teoría de la educación, sino también en filosofía e historia. Un libro de J. Huizinga (publicado originalmente en 1944, cuando era profesor de historia en la Universidad de Leyden, Alemania) ofrece un sólido argumento en el sentido de que el juego no es sólo un proceso de aprendizaje, sino que posee un valor sustancial a la cultura humana. La obra, titulada *Homo ludens* (derecha), resalta varios puntos sobre la importancia del juego:



- Es una función significativa: tiene un sentido
- Todo juego significa algo: dista mucho de ser una actividad inútil
- Es a la vez voluntario e instintivo. Los niños juegan instintivamente a aprender acerca del mundo que los rodea, pero siguen jugando porque les divierte
- La civilización contemporánea adolece de la formulación del juego: cuanto más estructurado sea éste, menos lúdico es

En la granja



Geoff Rowley

"Farmfax" es un juego de programas que cubre cabalmente los principales aspectos de una granja moderna

Partiendo del hecho de que los ordenadores se están utilizando en todos los sectores de la sociedad y de la empresa, no sorprende en absoluto que se haya creado una selección de programas para granjeros. Lo que sí puede resultar sorprendente es que la mayoría de estos programas están destinados a una máquina que estuvo a punto de zozobrar, el Dragon, que ahora se fabrica en España y está adquiriendo gran popularidad entre los agricultores.

Uno de tales programas es la serie Farmfax, cuyo creador es el granjero Geoffrey Paterson. La aplicación práctica de cada paquete está orientada hacia una tarea específica de la granja, y la gente que escribió las especificaciones y las tradujo al lenguaje assembly sabía muy bien lo que hacía. Los programas están diseñados para ser fácilmente comprendidos por personas que no estén familiarizadas con los ordenadores y se basan en cartuchos de ROM que quedan listos para utilizar tras ser enchufados. Algunos de ellos poseen, asimismo, RAM extra en el cartucho para un almacenamiento de datos fácil y rápido, aunque es aconsejable que los usuarios conserven copias por razones de seguridad.

Un ejemplo de cómo el granjero puede sacar partido de la inclusión del micro en la administración de la granja es Robin Dunkerley, que administra una granja de 500 acres en las afueras de Oxford. Aproximadamente la mitad de sus tierras están reservadas para su rebaño de 150 Holsteins y Friesians, quedando la mayor parte del resto para el cultivo. Uno de los problemas que el programa Farmfax le ha ayudado a resolver es la determinación del "Índice de Calving" (el número de días que pasa una vaca sin parir y continuando, por consiguiente, su producción de leche). "Se ha comprobado que si una vaca no pare al menos una vez al año —explica—, a uno le cuesta unas 2 libras esterlinas por cada día de los 365 que ha estado sin parir. En el pasado, se nos han pasado 390 días o más; el promedio ha sido de 375. El ordenador nos ayuda a seguir el rastro de cualquier vaca que no haya sido entregada al toro y hasta ahora no nos ha fallado ni uno. El año pasado nuestro índice de Calving fue sólo de 366, y tengo mis dudas de si llegaremos alguna vez a conseguir uno mejor. Hemos alcanzado casi un índice óptimo."

Como indicador del ámbito de un programa, el *Individual cow records* puede retener todos los registros de hasta 240 cabezas de ganado dentro del cartucho; una versión más económica los almacena en cinta. Puesto que todos los datos se cargan en la memoria y se manipulan antes de volver a guardarse (SAVE), ambas versiones trabajan de forma idéntica. Sin embargo, aunque muchos considerarán la cinta como un medio más lento para el almacenamiento de datos, a los granjeros el tiempo les resulta bien invertido, ya que con anterioridad debían realizar tediosos cálculos para adoptar decisiones tales como cuándo vender una vaca cuya producción está disminuyendo y comparar lo que está costando en alimentos con lo que está produciendo en venta de leche. Y, lo que es más significativo, encuentran que es más eficaz que alquilar por horas un ordenador central.

"Solíamos pagar unas libras mensuales a un centro de cálculo para conservar nuestros registros en su ordenador", recuerda Dunkerley. "A fin de mes les enviábamos la información, y las salidas impresas nos llegaban unas tres semanas después, cuando ya era demasiado tarde para actuar en función de la información. Asimismo, gran parte del material de

la información que nos proporcionaban era valioso para quien tuviera una rebaño de mucho pedigree, pero era totalmente irrelevante para un rebaño lechero comercial como el mío, que se prepara tan sólo para producir leche." Durante mucho tiempo pensó que sería totalmente lógico adquirir su propia máquina; pero, sin embargo, Dunkerley no estaba preparado para pagar 2 500 libras. Pero el Dragon, la impresora y los programas Farmfax, cuyo precio completo oscilaba entre 600 y 700 libras, sí estaba al alcance de su presupuesto. Su gasto se amortizó rápidamente y ahora tiene acceso a la información que necesita cuando lo requiere, en vez de tener que pagar por información que con frecuencia le llegaba tarde y era irrelevante.

Ahora, al final de cada mes, Dunkerley pasa alrededor de una hora organizando y calibrando su información. Primero ejecuta el *Dairy management program* (programa para administración de lecherías), que le da resultados financieros, el margen de ingresos por encima de los gastos de alimentación, sus ingresos brutos, cuántos kilos de forraje está utilizando para generar cada litro de leche, la producción total de leche y otros datos importantes. Después ejecuta el *Dairy ration* para asegurarse de que está produciendo la cantidad debida de leche. Le resulta muy útil porque lleva a cabo un análisis de alimentación completo. Entrando la cantidad de megajulios de energía y cuánta materia seca y proteína cruda digerible hay en cada kilo de alimento, y entrando el precio y si se ha producido o no en la granja, Dunkerley puede determinar el costo total y el costo de cada vaca por día.

Entrando la producción diaria de leche de cada vaca, el programa *Individual cow records* proporcionará un registro de absoluta precisión al cabo de cada mes. "Si cambio algunos de los factores —explica Dunkerley—, el ordenador me vuelve a calcular todo automáticamente. Esto es lo bonito de un ordenador: gracias a él, el tiempo que lleva esta clase de trabajo se reduce a menos de un tercio. Y también me informa de cosas que a mí siempre me había interesado saber, por ejemplo, cómo marcha mi giro en descubierto bancario. Yo, como agricultor arrendatario, soy muy vulnerable al fatídico día, que se produce dos veces al año, en que vence mi alquiler. También puedo hacer un presupuesto anticipado exacto, lo que satisface en gran medida al gerente de mi banco."

Cada uno de los módulos del programa Farmfax opera sobre una base de gran amabilidad hacia el usuario, e incluye un manual, pequeño y simple, cuyo estudio sólo requiere unos minutos, tratándose, por lo general, de una hoja de papel plegada que cabe en el bolsillo. Todos ellos empiezan con un sencillo menú. Por ejemplo, *Cullcow,* que ayuda al granjero en aquellas decisiones relativas a considerar las vacas cuya producción de leche cae por debajo de un cierto nivel, primero le solicita al granjero que escoja entre:

```
PAGE 1 — DETALLES DE PRECIO
PAGE 2 — MEJOR COMPRA/VENTA
```

En la página 1, el granjero debe entrar primero el mes y el año a considerar, luego el tipo de interés bancario aplicable, el precio de la leche, la producción de leche en litros por día de cada vaca, y el costo mensual. Apenas se entran estos cuatro elementos, las mismas cifras se transfieren a todas las otras entradas para ahorrar tiempo de digitación, si bien se las puede alterar de forma manual si se piensa, por ejemplo, que las tasas de interés pueden variar en los meses siguientes.

En la página 2 se le pregunta al granjero si está comprando o vendiendo vacas, la cuantía del lote, el precio esperado y el mes de venta. El ordenador compara entonces todas las cifras y coloca una estrella junto a la vaca que representa la mejor oferta. Pulsando la tecla BREAK en cualquier momento el usuario regresará al índice, para volver a entrar cualquier dato si así fuera necesario.

Los otros paquetes operan sobre una base similar, si bien, como es obvio, en los programas más complejos el menú de apertura será más complejo. Por ejemplo, la página de apertura del *Dairy ration formulation* ofrece:

```
PAGE 1 — TABLA DE ALIMENTOS
     2 — MINERALES
     3 — COMPARABILIDAD
     4 — DESCRIPCION DE GRUPOS
     5 — FORMULACION
     6 — CANTIDAD DE PIENSOS
     7 — ARCHIVO/RECUPERACION
```

Si se emplea un cartucho sin memoria, el programa se activa automáticamente en la opción 7, permitiendo recuperar desde cinta datos anteriores; de lo contrario, se visualiza el menú. Cada uno de los alimentos entrados junto con sus precios en la página 1 se repite de forma automática en la página 2, en la cual el usuario puede entrar el contenido de minerales de cada uno, mientras que en la página 3 se analizan los datos, produciendo una evaluación de cuánta energía, cuánta materia seca y proteína suministra cada una por cada libra esterlina. Si el granjero entra en la página 4 sus objetivos de producción en términos de peso de la vaca, días de estar preñada, litros de leche producidos y contenido de grasa de la leche, entonces se puede, con la ayuda del programa para formular su proporción alimenticia, igualar al instante los costos totales de alimentación para un número dado de vacas durante una cantidad dada de días.

Evidentemente, ésta es la clase de programas escritos por especialistas y para especialistas. Los granjeros como Robin Dunkerley valoran la asistencia que este software ofrece en un mundo agrícola cada vez más industrializado y agobiado por la burocracia administrativa. "Con los cupos de la CEE y las presiones intensificándose cada vez más —concluye—, nos vemos obligados a llevar la granja de la forma más controlada posible, y el ordenador nos ayuda a hacerlo. No estoy seguro de si efectivamente nos ayuda a ganar más dinero, pero, por cierto, sí nos ahorra muchísimo."

Farmfax Suite: Para el Dragon
Distribuido por: Farmfax Computer Systems, PO Box No. 2, Stockbridge, Hampshire SO20 6LE, Gran Bretaña
Autor: Geoffrey Paterson
Formato: Cartucho

Ian McKinnell

	(1)	(2)	(3)	(4)	
U$()	1 Kilo	1 Kilo	1 Kilo		UNIDADES
P$()	Verduras	Fruta	Carne	Agua	DESCRIPCIONES
PC()					COSTES
PN()	0,5	1	1	0,5	PROVISIONES MÍNIMAS POR HOMBRE Y SEMANA
PA()	300	200	100	200	CANTIDAD DE PROVISIONES PEDIDAS

Necesidades alimenticias
El jugador puede seleccionar cuatro tipos de provisiones: verduras, fruta, carne y agua. Las descripciones de estos cuatro tipos están almacenadas en la matriz P$() y las unidades correspondientes en la matriz U$(). Para tomar las decisiones relativas a la cantidad de provisiones a adquirir, el jugador debe disponer de dos clases de información: el coste de cada provisión y las necesidades mínimas por semana por cada miembro de la tripulación. Estos datos están retenidos en las matrices PC() y PN(). Estas cuatro matrices se establecen al principio del programa. Hay otra matriz que se va llenando a medida que se encargan provisiones. PA() retiene la cantidad de cada provisión

Líneas de suministro

Nos corresponde desarrollar el segundo módulo del programa para el juego mercantil "Nuevo Mundo"

Al pulsar cualquier tecla aparece en pantalla la segunda etapa, en la que deben adquirirse las provisiones para la travesía. El jugador sabe que el viaje durará alrededor de ocho semanas, de modo que habrá que llevar una adecuada cantidad de alimentos y de agua.

El programa calcula las necesidades totales para ocho semanas y comprueba que se hayan adquirido provisiones suficientes. De no ser así, avisa al jugador, diciéndole que alguien podría padecer hambre o sed, y le ofrece la oportunidad de efectuar otro pedido. Si el jugador no dispone de oro suficiente para abonar el pedido, entonces el programa le indica que vuelva a probar (éste visualiza un balance de dinero durante la etapa de aprovisionamiento).

Al igual que en la etapa de contratación, necesitamos preparar algunas matrices:

```
19 REM **** MATRICES DE APROVISIONAMIENTO ****
20 DIMU$(4):U$(1)="KILO":U$(2)="KILO":U$(3)="KILO":U$(4)="BARRIL"
21 DIMP$(4):P$(1)="VEG":P$(2)="FRUTA":P$(3)="CARNE":P$(4)="AGUA"
22 DIMPA(4)
23 DIMPC(4):PC(1)=.5:PC(2)=1:PC(3)=2:PC(4)=.5
24 DIMPN(4):PC(1)=2:PN(2)=1:PN(3)=1:PN(4)=.5
```

Las provisiones se suministran en unidades diferentes: las verduras, la fruta y la carne se miden por kilos y el agua por barriles. La línea 20 prepara una matriz, U$, con cuatro elementos para retener estas cuatro unidades. Grabar las unidades en una matriz en vez de tener que imprimirlas continuamente representa un ahorro de memoria. Si desea ampliar la lista para incluir sacos de harina, por ejemplo, esta matriz se puede ampliar fácilmente.

Las descripciones de los cuatro tipos de provisiones están representadas en la línea 21 por los cuatro elementos de la matriz P$(). El programa prepara la matriz PA() para almacenar la cantidad adquirida de cada provisión. Un kilo de verduras cuesta media pieza de oro, un kilo de fruta una pieza de oro, un kilo de carne dos piezas de oro y un barril de agua media pieza de oro. Esta información se

almacena en la matriz de la línea 23. El orden de las provisiones no se enuncia en esta línea, sino que se establece por defecto. La matriz se puede ampliar fácilmente para incluir provisiones extras.

Para mantener un buen estado de salud, cada miembro de la tripulación necesita cierta cantidad de cada provisión. Estas necesidades se establecen en una matriz, PN(), en la línea 24. Por ejemplo, un miembro de la tripulación necesita dos kilos de verduras por semana, de modo que el primer elemento, PN(1), se establece en 2.

La línea 500 del primer módulo del programa envía el programa a la subrutina de la línea 1000 para contratar la tripulación. Tras la contratación, retorna a la línea 500. Para llamar a la rutina de aprovisionamiento hemos de insertar una llamada GOSUB después de la línea 500; de este modo:

```
550 GOSUB 2000: REM APROVISIONAMIENTO
```

Se visualiza una pantalla que explica el aprovisionamiento. La variable CN, establecida durante la etapa de contratación, representa el número de tripulantes contratados. Se utiliza en la línea 2040 para recordar cuántas bocas se han de alimentar.

La programación del aprovisionamiento se halla dentro de un bucle y se emplea cuatro veces, una vez para cada tipo de provisión. El valor del subíndice T se incrementa cada vez en uno. Éste se prepara en la línea 2050. La pantalla imprime la cantidad de cada tipo de alimento que necesita cada tripulante por semana. Estas necesidades se determinan usando el elemento correspondiente de PN().

Como hemos visto, los alimentos se miden por kilos y el agua por barriles. Las unidades requeridas se determinan mediante el elemento correspondiente de la matriz U$(), definida en la línea 20. Si se requiriera más de un kilo, la línea 2075 le añade a kilo una S; de ir en singular, la línea 2080 reemplaza la S por un espacio. La línea 2085 imprime DE y el nombre de la provisión retenida en P$(), seguido de POR CADA SEMANA QUE DURE EL VIAJE.

En las líneas 2100 y 2110 se le pregunta al jugador cuántas unidades de cada provisión desea comprar. Las demandas se digitan y se representan mediante P$() en la línea 2130. La línea 2140 coloca la cantidad en el primer elemento de la matriz de cantidades. Si la cantidad pedida es cero, el programa imprime SI NO COMPRAS NADA DE seguida de la matriz de provisiones ;P$(T) en la línea 2160. El programa calcula entonces si la cantidad pedida y entrada en la matriz PA() es mayor que la requerida para un viaje de ocho semanas, utilizando la fórmula CN*8*PN(T). CN es el número de tripulantes, y PN(T) corresponde a las necesidades de una provisión dada, determinada por T. Esto se produce en la línea 2170. Si las provisiones son insuficientes se imprime un mensaje de advertencia. El programa comprueba si está realizando el bucle por cuarta vez. De ser así, imprime SED en vez de HAMBRE.

Si se le ha advertido al jugador que las provisiones son insuficientes, el programa (en la línea 2230) le dice que: QUIERES PROBAR OTRA VEZ (S/N)? Si el jugador, a pesar de disponer de pocos fondos, acepta la entrada, el programa pasa a la línea 2400. Si desea hacer un pedido diferente, el valor pertinente de la matriz PA() se establece en cero. En este caso el programa ha de retroceder hasta el mismo tipo de provisión. Ello lo hace restándole 1 al contador del bucle mediante T=T−1, en la línea 2250.

Si el jugador está satisfecho con el pedido realizado, entonces se visualizan la cantidad de dinero que le queda y una lista de las provisiones pedidas hasta el momento. Puede ser que el jugador haya comprado provisiones suficientes para el viaje, pero que no le quede oro para abonar la factura. La línea 2260 comprueba si es éste el caso. De ser así, las líneas 2270-2290 indican que no hay dinero suficiente para comprar la cantidad de provisiones y se le solicita al jugador que vuelva a probar. Si la compra fue válida, su costo se resta del oro disponible, retenido en MO.

En la línea 2410, dentro de otro bucle controlado por TT, se imprime una lista de las provisiones adquiridas hasta el momento, utilizando las variables U$(), PA() y P$(). La cantidad de oro que aún queda se visualiza en la línea 2480. El bucle termina en la línea 2490. El jugador pasa a la siguiente etapa respondiendo al mensaje PULSE CUALQUIER TECLA PARA CONTINUAR. En este punto, el control retorna al programa principal.

Módulo dos

```
 000 REM **** APROVISIONAMIENTO ****
2005 PRINTCHR$(147):REM LIMPIAR PANTALLA
2010 S$="       ETAPA 2 - APROVISIONAMIENTO*"
2015 GOSUB 9100
2020 S$="       -------       ----------------*"
2025 GOSUB 9100
2030 GOSUB 9200:PRINT
2040 PRINT"HAS CONTRATADO UNA TRIPULACION COMPUESTA POR
     ";CN;"MIEMBROS."
2045 GOSUB 9200:GOSUB 9200
2050 FOR T=1 TO 4
2055 PRINT
2060 PRINT"CADA MIEMBRO DE LA TRIPULACION NECESITARA "
2070 PRINT"AL MENOS ";PN(T);" ";U$(T);
2075 IFPN(T)=1THENPRINT" ";:GOTO2085
2080 PRINT"S";
2085 PRINT"DE";P$(T)
2086 PRINT"A";PC(T);"PIEZAS DE ORO POR ";U$(T)
2090 PRINT"POR CADA SEMANA QUE DURE EL VIAJE."
2095 GOSUB 9200:PRINT:GOSUB 9200
2100 PRINT"CUANTOS ";U$(T);"S DE ";P$(T)
2110 S$="DESEAS COMPRAR*":GOSUB 9100
2120 PRINT
2130 INPUTP$
2140 PA(T)=VAL(P$):GOSUB 9200
2150 IFPA(T)>((CN*8*PN(T))-1)THEN2260
2160 IFPA(T)=0THENPRINT"SI NO COMPRAS NADA DE":GOTO2180
2170 PRINT"SI SOLO COMPRAS ";PA(T);U$(T);
2175 IFPA(T)=1THENPRINT" DE":GOTO2180
2176 PRINT"S DE"
2180 PRINTP$(T);",":GOSUB 9200
2190 PRINT"ALGUIEN PODRIA PASAR ";
2200 S$="HAMBRE"
2210 IF T=4 THEN S$=SED"
2220 PRINT S$;"!":GOSUB 9200
2230 S$="QUIERES VOLVER A PROBAR (S/N)":GOSUB 9100
2240 INPUTP$:P$=LEFT$(P$,1)
2242 IF P$ <> "S" AND P$ <> "N" THEN 2230
2245 IF P$="N" THEN 2400
2250 PA(T)=0:T=T-1:GOTO2410
2260 IFPA(T)*PC(T)> MOTHEN2270
2265 GOTO 2400
2270 S$="NO TIENES DINERO SUFICIENTE PARA":GOSUB 9100
2280 PRINTPA(T)
2290 PRINTU$(T);"S DE";P$(T):GOSUB 9200
2300 S$="POR FAVOR VUELVE A PROBAR*":GOSUB
     9100:PA(T)=0:T=T-1:GOTO 2410
2400 MO=MO-(PA(T)*PC(T))
2410 PRINT:S$="PROVISIONES HASTA AHORA:*":GOSUB 9100
2412 GOSUB 9200
2415 FOR TT=1 TO 4
2420 PRINT PA(TT);U$(TT)
2430 IF PA(TT)=1THEN PRINT" DE ";:GOTO2440
2435 PRINT"S DE ";
2440 PRINTP$(TT)
2450 GOSUB 9200
2460 NEXT
2480 PRINT"DINERO QUE TIENES AUN = ";MO;"PIEZAS DE ORO"
2485 GOSUB 9200:GOSUB 9200
2490 NEXTT
2500 GOSUB 9200:PRINT:S$="FIN DEL APROVISIONAMIENTO*":GOSUB 9100:
     GOSUB 9200
2510 PRINT: S$=K$:GOSUB 9100:PRINT: GOSUB9200
2520 GETP$:IFP$=""THEN2520
2999 RETURN
```

Líneas de abastecimiento
Esta subrutina se ocupa del aprovisionamiento para el viaje y debe ser añadida al primer módulo, que hemos ofrecido anteriormente. El programa pasa cuatro veces por un bucle, permitiendo que el jugador encargue una de las cuatro provisiones en cada pasada

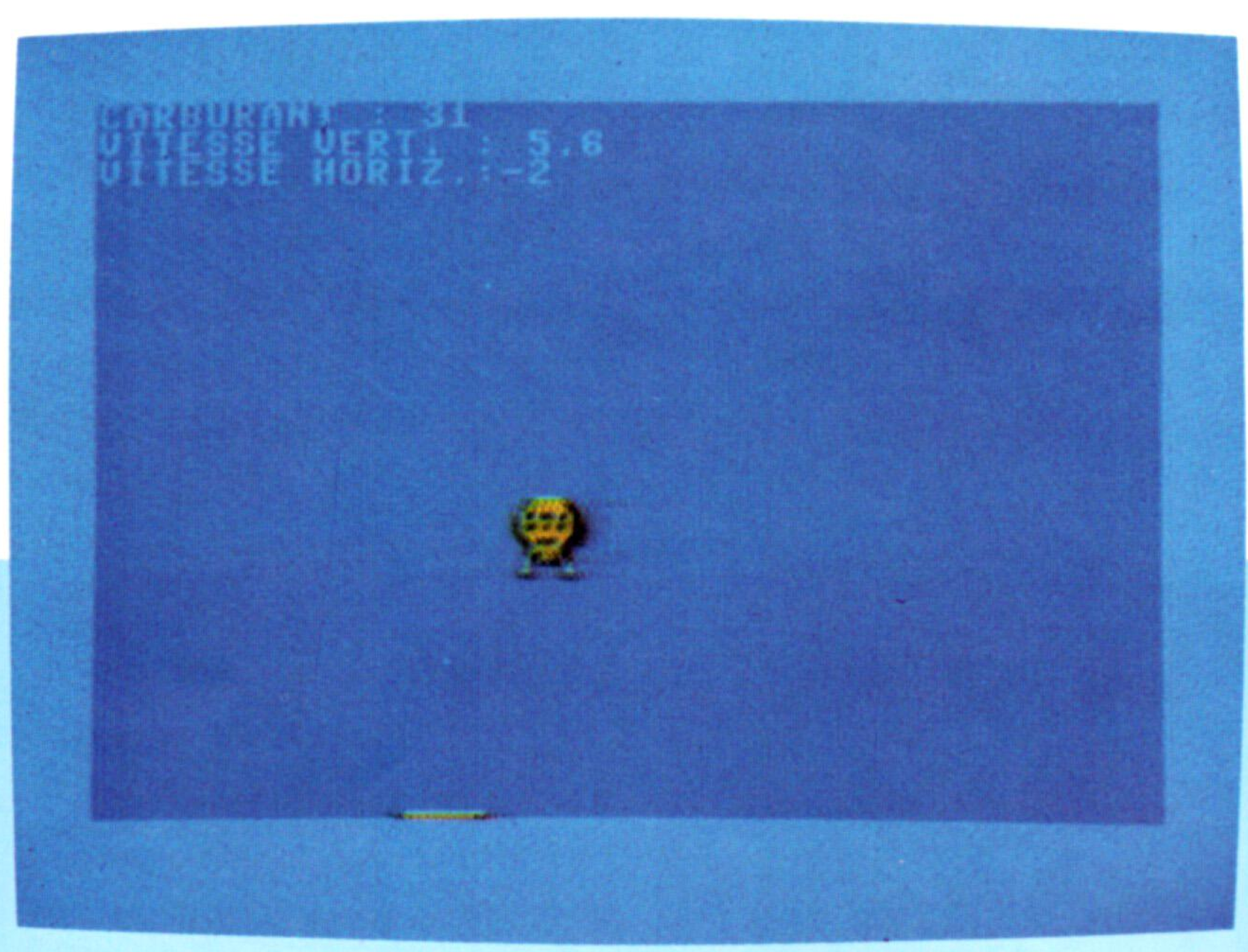
Complementos al BASIC

Spectrum:
Introduzca las siguientes modificaciones:
```
2005 CLS
2240 INPUT P$:LET P$=
P$(1 TO 1)
```

BBC Micro:
Introduzca las siguientes modificaciones:
```
2005 CLS
```

Alunizaje

Esta versión es para el Commodore 64. Un consejo: grabe el juego en cassette para evitar molestias



Después de un largo viaje sin gravedad, no resulta nada fácil alunizar suavemente con una nave espacial; pero gracias a su ordenador, está en condiciones de efectuar un entrenamiento sin riesgos. Debe colocar su nave sobre la zona destinada al efecto. Puede dirigirse hacia la derecha y hacia la izquierda con ayuda de las teclas de control del cursor, o bien frenar el descenso mediante la barra espaciadora. Para conseguir el alunizaje, las velocidades verticales y horizontales deben ser iguales o inferiores a 1.

```
5 REM *************
10 REM * ALUNIZAJE *
15 REM *************
20 GOSUB 1000
100 GET X$
105 IF FU=0 THEN 150
110 IF X$<>"" THEN FU=FU-1
120 IF X$="[1SPC]" THEN VV=VV-1
130 IF X$=G$ THEN VH=VH-1
140 IF X$=D$ THEN VH=VH+1
150 VV=VV+0.1
160 V=V+VV
180 H=H+VH
190 IF H>255 THEN H=0:MS=M1
200 IF H>90 AND MS=M1 THEN 4540
210 IF H<0 AND MS=M1 THEN H=255:MS=MO
220 IF H<0 AND MS=MO THEN 4540
240 PRINT HO$;
250 PRINT "CARBURANTE[1SPC]:";STR$(FU); "[4SPC]"
255 IF V<0 THEN 300
260 POKE D+16,MS
265 POKE D,H
270 POKE D+1, INT(V)
275 IF V>V1-3 THEN 3000
300 PRINT "VELOCIDAD[1SPC]VERT.[1SPC]:";STR$
    (INT(10*VV)/10);"(4SPC]"
310 PRINT "VELOCIDAD[1SPC]HORIZ.:";STR$(VH);
    "[4SPC]"
320 GOTO 100
1000 D=53248
1005 RESTORE
1010 FOR I=0 TO 190
1020 READ A
1030 POKE 832+I,A
1035 NEXT I
1040 M1=5
1050 M0=0
1060 POKE D+39,7
1080 POKE D+40,7
1090 POKE D+41,0
1100 POKE 2040,13
1110 POKE 2041,14
1120 POKE 2042,15
1130 V1=230
1140 V0=0
1150 FU=90
1160 G$=CHR$(17)
1170 D$=CHR$(29)
1180 HO$=CHR$(19)
1190 M=54272
2000 PRINT CHR$(147);
2010 POKE D+21,0
2020 H=INT(RND(TI)*191)+65
2030 AH=INT(RND(TI)*191)+65
2050 FU=FU+10
2060 MS=M0
2070 VH=0
2080 VV=0
2090 V=0
2100 POKE D+2,AH
2110 POKE D+3,V1
2120 POKE D,H
2130 POKE D+1,V
2140 POKE D+21,3
2150 RETURN
3000 IF ABS(H-AH)>4 THEN 4000
3005 IF VV>1 THEN 4000
3010 IF ABS(VH)>1 THEN 4000
3015 FOR I=1 TO 4000
3020 NEXT I
3030 SC=SC+1
3040 GOSUB 2000
3050 GOTO 100
4000 POKE D+5,V+5
4010 POKE D+4,H
4020 POKE D+21,6
4500 FOR I=1 TO 5
4510 PRINT
4520 NEXT I
4530 PRINT TAB(6)"LA[1SPC]NAVE[1SPC]SE[1SPC]
     HA[1SPC]ESTRELLADO"
4540 IF SC>RE THEN RE=SC
4545 FOR I=1 TO 2000
4550 NEXT I
4555 FOR I=1 TO 3
4560 PRINT
4570 NEXT I
4580 PRINT TAB(13)"PUNTOS[1SPC]:";SC
4590 SC=0
5000 FOR I=1 TO 3
5010 PRINT
5020 NEXT I
5030 PRINT TAB(10)"PUNTUACION[1SPC]MAXIMA
     [1SPC]:"RE
5040 IF RE<SC THEN RE=SC
5050 SC=0
5060 FOR I=1 TO 3
5070 PRINT
5080 GET X$
5090 NEXT I
5100 PRINT TAB(13)"OTRA[1SPC]?"
5110 GET X$
5120 IF X$="" THEN 5110
5130 IF X$()"N" THEN POKE D+21,0:GOTO 20
5140 END
10000 DATA 0,255,0,1,255,128,3,255,192
10010 DATA 7,255,224,12,195,48,12,195,48
10020 DATA 15,255,240,12,102,48,12,102,48
10030 DATA 15,255,240,7,255,224,3,129,192
10040 DATA 1,129,128,0,255,0,1,255,128
10050 DATA 1,24,128,2,60,64,2,36,64
10060 DATA 4,0,32,4,0,32,14,0,112,0
10100 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
10110 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
10120 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
10130 DATA 0,0,0,0,0,0,0,0,0,0,255,255,255
10140 DATA 255,255,255,255,255,255,0
10200 DATA 0,0,0,0,0,45,0,27,3,0,0,8,0,127,0
10210 DATA 0,5,0,0,236,0,0,67,0,2,64,240
10220 DATA 7,32,112,7,32,56,15,145,56
10230 DATA 31,23,120,31,25,124, 95, 97, 254
10240 DATA 127,255,254,255,255,255,255
10250 DATA 255,255,0,0,0,0,0,0,0,0,0
```



Accesorio Acorn

El módulo de ampliación Plus 3 de Acorn Electron eleva el equipo al nivel requerido por usuarios serios de micros

Una de las críticas más serias que se le han hecho al Acorn Electron ha sido su falta de interfaces. Si bien la máquina utilizaba el veloz BASIC BBC, el hecho de que ésta sólo contara con un par de conectores para pantalla, un enchufe RF y una interface para cassette (no proporcionaba siquiera una puerta para palanca de mando) significaba que su potencial de ampliación estaba severamente limitado. Acorn siempre prometía módulos de ampliación que permitirían que los usuarios elevaran sus máquinas al nivel del BBC Micro, desde el punto de vista de interfaces disponibles, pero estos accesorios tardaban mucho en llegar.

El primer módulo de ampliación que salió al mercado fue el Plus 1, que ya hemos tenido ocasión de examinar, que contribuyó mucho a acallar las críticas que se le hacían al Electron. Pero aunque proporcionan muchas de las interfaces de que no disponía el Electron original, el Plus 1, sin embargo, no incluyó una interface para disco que permitiera el fácil acceso al almacenamiento masivo de datos.

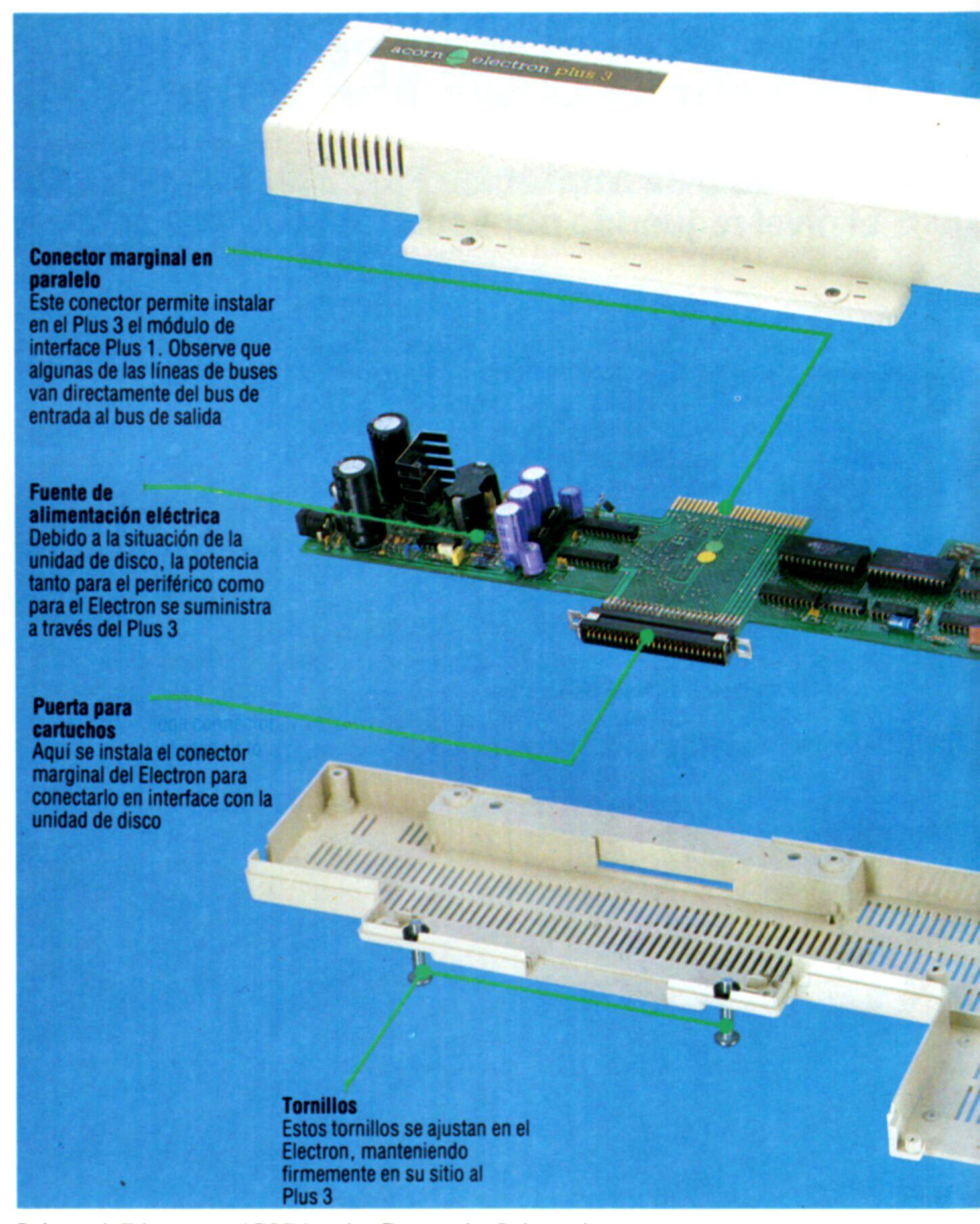
Ahora esta situación ha cambiado gracias a la aparición del Plus 3, un sistema de disco "todo en uno" para el Electron. El módulo tiene forma de "L", al instalarse el sistema de archivo en discos a lo largo de la parte posterior del ordenador, con la unidad de disco, más corta, colocada sobre el lado derecho; esta última, lamentablemente, tapa la fuente de alimentación eléctrica del Electron. El Plus 3, en consecuencia, viene con su propia fuente de alimentación eléctrica, de 36 V, que se enchufa en el lado izquierdo de la unidad y alimenta tanto a la unidad de disco como al Electron. Al igual que el Plus 1, el Plus 3 se desliza en el conector marginal de la parte trasera del ordenador y, para evitar que la máquina se mueva y pueda romper el conector marginal, el módulo se fija firmemente en la base de ésta mediante dos tornillos. Una vez instalado, el Electron y el Plus 3 conforman una sólida unidad, muy difícil de romper. A diferencia del Plus 1, el nuevo modelo posee su propio conector marginal instalado atrás, para permitir la adición del Plus 1 y de las unidades que pudieran aparecer en el futuro.

Lo que tal vez resulte sorprendente es que Acorn haya optado por dotar al Plus 3 de unidades de disco Sony de 3 ½ pulgadas, en vez del formato usual de unidades de disco de 5 ¼ pulgadas. Situado directamente detrás de la unidad de disco hay, asimismo, un segundo conector marginal que permite la adición de una segunda unidad de disco, la cual según Acorn utilizará no sólo unidades de 3½



Chris Stevens

Bloques armónicos
El accesorio Plus 3 de Acorn para el Electron está diseñado para acomodar también al Plus 1. La provisión de estas interfaces permite que el usuario del Electron mejore su ordenador, alcanzando unas especificaciones que se aproximan a las del BBC Micro (con unidad de disco) a un precio muy razonable



pulgadas, sino también de 5 ¼ pulgadas. Si el usuario desea agregar una unidad de disco de 5 ¼ pulgadas para el BBC Micro, Acorn ofrece una interface accesoria para poder instalar la unidad en el conector marginal.

En el interior del dispositivo, el Plus 3 se divide en cuatro partes. Comenzando por la izquierda, está el sistema de circuitos de la fuente de alimentación eléctrica, caracterizado por sus grandes condensadores. A continuación se hallan los buses que salen del Electron, algunos de los cuales conducen a la unidad de disco, y otros vuelven directamente al conector marginal externo que se conecta con el Plus 1. A continuación viene el sistema de circuitos del sistema de archivo en disco. Aquí los chips contienen las instrucciones extras basadas en ROM que permiten que uno le "hable" a la unidad de disco y los chips de gestión de los discos. Por último, a la derecha está la propia unidad de disco de doble cara, revestida en metal. Esta carcasa tiene un doble papel. Básicamente cumple el papel de disipador, para impedir que la temperatura del interior del módulo se eleve excesivamente, pero también actúa como caja de Faraday, impidiendo que los campos magnéticos dispersos interfieran en el sistema.

Los discos Sony están alojados en un plástico rígido, con una cubierta protectora en la ventana de lectura/escritura. Al formatearse, el disco se divide en 80 pistas de 16 sectores. Puesto que cada sector puede contener hasta 256 bytes de información, la capacidad total de un disco de una sola cara es de 320 Kbytes.

Cuando se instala el Plus 3 y se enciende la máquina, junto con el encabezamiento normal se visualiza el mensaje ACORN ADFS. Ésta es la sigla de Advanced Disk Filing System. Aunque el Electron DFS es muy similar en muchos sentidos al sistema del BBC Micro (muchas de las instrucciones son idénticas), existen sutiles diferencias en la operación del sistema. Con el Plus 3 se proporciona un disco Welcome que contiene programas de demostración similares a los de la cassette que viene con el Electron. Sin embargo, el disco también contiene numerosas utilidades de incalculable valor para la operación del sistema de archivo en disco. Entre las utilidades que se proporcionan se incluyen BACKUP, que permite copiar discos enteros, DIRCOPY, que permite copiar de un disco a otro sólo archivos específicos, BUILD, que permite incorporar al disco rutinas BOOT, y DUMP, que visualiza el contenido de un archivo en formatos hexadecimal y ASCII.

Al igual que el BBC Micro DFS, las instrucciones de disco del Electron van precedidas por la instrucción *, de modo que instrucciones del OS como *LOAD, *RUN y *DELETE les resultarán familiares a quienes hayan utilizado el sistema de archivo en disco del BBC Micro. La catalogación (CAT) del disco producirá una visualización idéntica a la del Micro, con el título del disco, el número de unidad, las opciones, el nombre del directorio actual y el nombre de la biblioteca actual con los archivos, listados debajo.

La diferencia fundamental entre los dos sistemas operativos es el método que emplean para cargar discos nuevos. Cuando se inserta el disco en la unidad del Plus 3, el usuario ha de informar al sistema operativo digitando la instrucción *MOUNT, que tiene el efecto de cargar en la memoria el Currently Selected Directory (CSD) y la Currently Selected Library (CSL). Cuando el usuario desea cambiar de disco, debe entrar la instrucción *DISMOUNT. Esta instrucción cierra todos los archivos secuenciales que pudiesen estar abiertos e inhabilita el CSD y la CSL. Una vez cambiado el disco en la unidad, se vuelve a impartir la instrucción *MOUNT. Exceptuando la adición de estas instrucciones, el sistema operativo de disco parece ser idéntico al del BBC Micro.

El ADFS está organizado por una serie de jerarquías de archivo. Ello permite el acceso fácil y rápido a los archivos, gracias a una serie de nombres de ruta (*pathnames*). En el extremo superior de la jerarquía se halla el directorio seleccionado actualmente. Tras el encendido, éste es por defecto el directorio raíz $. Para ir del CSD a cualquier otro directorio disponible en el disco actual, el usuario simplemente digita *DIR seguido del nombre de ruta elegido. Debajo del CSD puede haber una cantidad de archivos, o una cantidad de subdirectorios conocidos como *archivos de biblioteca*. Dado que el CSD sólo puede retener hasta 47 archivos al mismo tiempo, usted debe organizarlos en bibliotecas si es que desea retener en el disco una cantidad muy elevada. Estos subdirectorios pueden retener una cantidad de archivos que quizá no pudieran in-





Chris Stevens

```
>*CAT
$                         (06)
Drive:0                   Option 03 (Exec)
Dir. $                    Lib. "Unset"

!BOOT          LWR(02)    Help          DLR(03)
Library        DLR(04)    Welcome       DLR(05)
ZYSysHelp      L   (06)
>_
```

```
>*INFO *
!BOOT          LWR(02)    0000D4556     00000000
00000032       000040     000041
Help           DLR(03)    00008A
Library        DLR(04)    000125        00002000
Welcome        DLR(05)    00002000
ZYSysHelp      L   (06)
000038F0       000007
>_
```

Ian McKinnell

Pantallas del Acorn ADFS
El Acorn Advanced Disk Filing System para el Electron es similar al del BBC Micro. La pantalla superior muestra la visualización en respuesta a la instrucción *CAT. *INFO visualiza el nombre del archivo, sus permisiones de acceso, carga, ejecución y direcciones de disco, y la longitud del archivo

ACORN ELECTRON PLUS 3

DIMENSIONES

365 × 200 × 50 mm (unidad de disco);
365 × 90 × 50 mm (interface)

CAPACIDAD

320 K para un disco de una sola cara

INTERFACE

Enchufe para fuente de alimentación eléctrica, bus de entrada tipo cartucho, bus de salida de conector marginal

DOCUMENTACION

El manual que se entrega con el Plus 3 es exhaustivo y le ofrece al principiante un enfoque paso a paso para la utilización del dispositivo. No obstante, sería necesario corregir algunos detalles ínfimos

VENTAJAS

Por su precio, la interface es una buena inversión y el usuario obtiene al mismo tiempo un sistema de archivo en disco y una unidad de disco

DESVENTAJAS

Por el momento hay muy poco software disponible para la máquina. Además, para hacer un uso completo de la unidad de disco se necesitan extras tales como una interface para impresora, que sólo está disponible en el Plus 1

cluirse en el directorio CSD principal; por consiguiente, para acceder a un archivo llamado Pepe en un disco con un cierto número de directorios con bibliotecas, el formato sería $.library1.pepe. Aunque de entrada este método para acceder a un archivo puede parecer muy complicado, tiene la ventaja de permitir que cada una de las bibliotecas disponibles en el disco tenga un archivo llamado Pepe. Siempre que el usuario permanezca en la biblioteca adecuada, al volver a guardar (SAVE) el archivo éste no machacará los otros archivos Pepe retenidos en el disco.

Al igual que en el BBC Micro, en el ADFS del Electron se proporcionan numerosas utilidades de gran valor práctico. Hay, por ejemplo, una serie de instrucciones que permiten cambiar el nombre de discos, directorios y archivos. Hay disponibles, asimismo, varias instrucciones para manipular las permisiones de acceso a los archivos. Ello se realiza mediante la instrucción *ACCESS, que limita el acceso a un archivo. Por ejemplo, generando la instrucción *ACCESS file1 RL, se "bloquea" el fichero, evitando su borrado accidental, pero permitiendo su lectura. Por otro lado, impartiendo la instrucción *ACCESS file1 WR se puede leer el archivo, o escribir en él, con entera libertad. Para impedir que alguien copie del disco un archivo en código máquina, colocándole al archivo el sufijo E se limitarán la cantidad de instrucciones que se pueden utilizar en el archivo para ejecutarlo (*RUN) y también se limitarán los diversos métodos para borrarlo.

El manual que se incluye con el Plus 3 es muy amplio, incluyendo un curso de enseñanza completo, y responde al nivel de documentación propio de Acorn. Sin embargo, el manual tiene algunos ligeros descuidos que, aun siendo menores, resultan algo molestos. Por ejemplo, en la página 18 de la guía se mencionan las abreviaturas CSD y CSL, pero éstas no se explican hasta las páginas 22 y 26, respectivamente.

Desde que se lanzó el Electron, por lo general se lo ha considerado como el pariente pobre, de la familia Acorn, del BBC Modelo B Micro. La carencia de interface, en especial la omisión de una puerta para unidad de disco, llevó a muchos a la conclusión de que el Electron era poco más que una máquina de juegos que no se podía considerar como una buena inversión para aplicaciones serias. Sin embargo, con la reducción del precio del Electron y la aparición de las interfaces Plus 1 y Plus 3, ahora la máquina aparece como una propuesta mucho más interesante. Ahora el Electron se ha convertido en una fuerza a tener en cuenta incluso por los usuarios más exigentes.

Preámbulo

Iniciamos el diseño y construcción de una pequeña máquina capaz de asir y desplazar objetos

El análisis de las opciones de diseño por lo general comienza con la definición de la tarea que deseamos llevar a cabo. En nuestro proyecto, la tarea nominal es que la máquina sea capaz de desplazarse desde un punto de partida, recoger algo (supongamos, un objeto del tamaño y peso de un carrete de hilo de coser) y colocarlo en una caja, antes de retornar a su punto de partida, con un margen de 2,5 mm. Supondremos que todas las posiciones se conocen de antemano. Una extrapolación industrial de esta tarea es una máquina que empaquete artículos a medida que van llegando a través de una línea de producción.

¿Qué clase de máquina se precisaría para llevar a cabo esta labor? Evidentemente necesitaría varios motores. Estos motores podrían ser hidráulicos o neumáticos, pero sólo consideraremos un sistema de motores eléctricos.

De los tres tipos principales de motor eléctrico, el más simple es el tipo CD normal. Aunque tales motores son económicos, es difícil controlarlos con precisión desde un dispositivo digital, tal como un ordenador personal. Para obtener el grado de precisión requerido utilizando este tipo de motor, habríamos de añadirle codificadores de eje a los husillos del motor. Cada motor requeriría entonces una lógica decodificadora, mediante hardware o software, para permitir un control posicional exacto.

Los motores paso a paso y los servocontroladores de CD son mucho más fáciles de controlar por ordenador. Sin embargo, los motores paso a paso, como ya hemos visto, son caros, requieren toda clase de circuitos activadores y suelen ser demasiado grandes para nuestros fines. Para un sistema a pequeña escala como el proyectado, los servomotores ofrecen dos ventajas fundamentales: dado que se producen en forma masiva para el mercado del modelismo controlado por radio, se pueden conseguir con facilidad y son relativamente económicos.

En nuestro apartado ya hemos visto cómo trabaja un servomotor; en resumidas cuentas, un sistema de servomotor se compone de un pequeño motor CD de 5 V, controlado por un bucle de realimentación digital. La posición angular del husillo del motor se controla mediante un pequeño potenciómetro y se compara con la señal de instrucción del chip incorporado. La señal de instrucción es un impulso de 5 V de entre uno y dos milisegundos de duración, determinando la longitud del impulso el ángulo que abarca el husillo del motor. Los impulsos de instrucción deben repetirse una vez cada 50 Hz (ciclos por segundo).

En nuestra serie también hemos descrito el software activador para servomotores tanto simples como múltiples. El programa es activado por interrupciones, lo que permite que el software en BASIC se ejecute como tarea de fondo y se interrumpa a intervalos regulares para enviar a los motores una nueva serie de impulsos.

Puente transversal móvil

Brazo con junturas

Posibles prototipos
He aquí dos posibles diseños para máquinas que puedan recoger una bobina de hilo y colocarla en una caja o recipiente. El puente transversal móvil no puede realizar ciertas tareas más complejas de las que sí es capaz el brazo-robot. Ofrece la ventaja de una geometría de movimiento más simple; los motores del brazo-robot exigen un control mucho más complicado para obtener un movimiento lineal simple de la pinza. No obstante, el control por ordenador significa que estos cálculos complejos son fáciles de manejar

La geometría de la máquina

De inmediato surgen muchos diseños posibles para una máquina que lleve a cabo la tarea que hemos descrito, pero primero debemos considerar las limitaciones de diseño. La primera consideración es que para recoger un objeto como un carrete de hilo necesitamos un mecanismo de presión. Éste debe utilizar como mínimo un motor. En segundo lugar, la "pinza" se debe mover en tres planos. La máquina debe ser capaz de moverse al menos

Sistema de geometría compleja

Geometría compleja
El control por ordenador de motores eléctricos, hidráulicos y neumáticos facilita el desarrollo de máquinas de gran flexibilidad. La compleja interacción de los diversos motores que mueven las secciones de la máquina se puede manipular mediante software sofisticado, lo que, en cierta medida, queda oculto al usuario de la máquina. Piense en las complejas manipulaciones que permite este sistema de seis arietes, estando los seis arietes extensibles bajo el control de software desde un ordenador

10 cm de izquierda a derecha (el plano *x*), unos pocos centímetros hacia atrás y adelante (el plano *y*) y al menos 10 cm hacia arriba y abajo (el plano *z*). Cada plano de movimiento requerirá al menos un motor, y la pinza debe maniobrarse mediante una combinación de estos motores con una precisión de al menos 2,5 mm, para permitirle retornar a su punto de partida.

Asimismo, la pinza debe ser capaz de recoger un peso de al menos 20 g, y sus uñas deben ser capaces de abrirse al menos 3 cm para hacer lugar a un objeto del tamaño de una bobina de hilo.

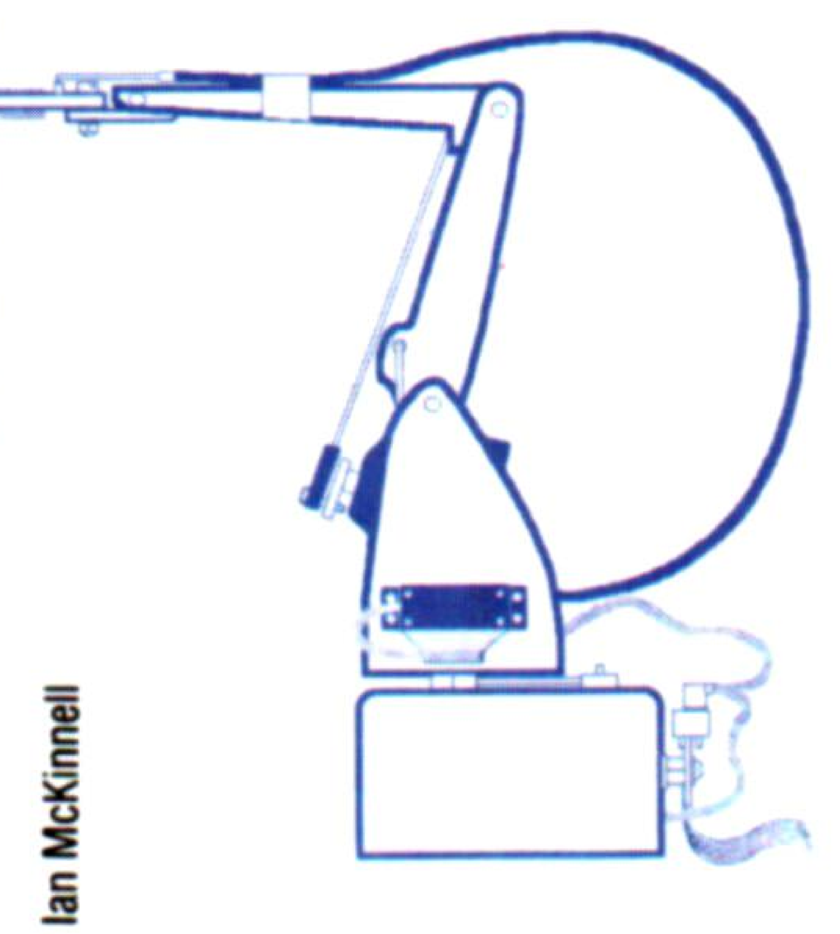
Un diseño que satisfaría estos requisitos sería un pequeño sistema de puente transversal basado en una estructura rectangular. El puente transversal y la pinza se podrían desplazar uniéndolas mediante caballetes a poleas de cable, que a su vez se podrían conectar a servomotores o motores paso a paso. Este sistema requeriría cuatro motores, tres para controlar el movimiento y uno para controlar la acción de prensión.

Una segunda posibilidad, y la que más se aproxima a la forma humana, es un brazo con junturas. La rotación de cada juntura se podría controlar mediante un solo motor. Los servomotores digitales son muy aptos para esta aplicación, dado que el ángulo del motor se dirige fácilmente mediante software para ordenador, permitiendo un control exacto de los miembros del brazo. El movimiento de la pinza mediante un brazo-robot requiere un control más complejo que efectuar el mismo movimiento mediante un puente transversal. Por ejemplo, para mover la pinza sólo en un plano (supongamos, el plano *x*) se requiere el control simultáneo de al menos dos motores del sistema de brazo con junturas, mientras que en el sistema de puente transversal este movimiento se podría conseguir utilizando sólo un motor. Sin embargo, dado que el sistema se controlará por ordenador, las secuencias de control complejas no ofrecen mayor problema: se pueden obtener mediante el uso de un software inteligente.

El brazo con junturas posee una ventaja respecto al puente transversal: es más flexible. Mientras que el puente transversal podría realizar la tarea específica que hemos esbozado, no podría realizar otras tareas sin una reacomodación física de los componentes del sistema. El sistema de brazo con junturas, sin embargo, podría efectuar modificaciones de la tarea establecida, tales como colocar un carrete de hilo en una caja que estuviera a su lado. En consecuencia, elegiremos el sistema de brazo-robot como base para nuestro proyecto de construcción.

En este primer capítulo proporcionamos un diagrama esquemático del diseño básico.

Color por números
En cada capítulo del proyecto iremos coloreando la sección pertinente de este esquema para indicar la parte del brazo-robot de la que nos estemos ocupando



Ian McKinnell

¿1+2=20?

En este capítulo analizaremos ciertos aspectos de las habilidades aritméticas del PASCAL

¿Cuánto es uno más dos? Sin más información, podríamos aventurar la respuesta: tres, por supuesto. El supuesto implícito es que estamos tratando con números naturales puros; pero supongamos que introducimos una moneda de 10 peniques y dos monedas de 5 peniques en una máquina tragaperras: el resultado sería 20 peniques (o, considerado desde un ángulo diferente, una taza de café y dos peniques de vuelta). Esta clase de clasificación de datos es fundamental para la descripción de cualquier problema, y el PASCAL nos ayuda a organizar nuestros datos y describirlos de una forma clara y lógica.

Como ya hemos visto, podemos crear categorías de datos nuevas y especiales, de modo que podamos pensar en la solución del problema y diseñar algoritmos basados en el proceso de cada objeto de datos de una forma apropiada a su tipo. Si inadvertidamente intentáramos hacer algo impropio, como leer del teclado un valor booleano, el compilador de PASCAL detectaría de inmediato este error lógico. Como comprenderá, ¡se pueden ahorrar muchísimas horas de frustrante trabajo de depuración si el compilador del lenguaje no lo deja siquiera ejecutar una sola instrucción hasta haber eliminado todos los errores del programa fuente! La incompatibilidad entre diferentes tipos de datos y la riqueza de las formas en las que podemos describirlos es una de las cualidades más valiosas del PASCAL. Lejos de sentirnos limitados en cuanto a lo que podemos hacer, con frecuencia desearemos aprovechar más las descripciones de datos fuertemente categorizadas en tipos del PASCAL e imponerles a las variables de forma deliberada algunas restricciones adicionales por nuestra propia cuenta.

Ahora que ya hemos utilizado todos los tipos de variables simples (aquellas que sólo pueden tener un valor), podemos resumir las reglas para manipularlas. Ninguna de las escalares es "compatible" con una variable de cualquier otro tipo escalar, si bien los dos tipos numéricos poseen muchas características en común y de vez en cuando se les permite "codearse" entre sí. Puesto que un número real se puede aproximar a un número entero, es posible la asignación de enteros a reales, pero jamás al contrario.

He aquí algunos ejemplos:

```
Program Compatibilidad (input,output);

VAR
   entA,
   entB :integer;
   Xreal,
   Yreal :real;

BEGIN
   read (entA, Xreal); {leer un entero, luego cualquier
   número legal}
Yreal:=entA; {real:=entero es OK}
   entB:=Xreal;{**ERROR: ilegal**}
   {..etc.
```

Las operaciones aritméticas sólo se definen aplicadas a los tipos numéricos, lo que no es del todo sorprendente. Tanto para los enteros como para los reales, se pueden utilizar los cuatro "operadores" simbólicos usuales:

+ suma
− resta
* multiplicación
/ división con punto flotante

En este contexto, son operadores *diádicos* o binarios, dado que siempre exigen dos "operandos" de cualquiera de los tipos numéricos. Cuando uno de los operandos es real, el resultado de la expresión es real: de modo que 2+2.0 es 4.0 (no 4). En el caso del signo de división, la expresión da un valor numérico real cuando ambos operandos son enteros: 3/5 es 0.6, 8/4 es 2.0.

Cuando dividimos valores integrales, con frecuencia una respuesta "real" no tiene sentido. Doce chocolatinas divididas entre diez personas da una para cada una y dos chocolatinas sobrantes, por ejemplo. El resultado entero y el resto se pueden obtener mediante los dos operadores de división entre enteros, DIV y MOD (que en PASCAL son ambos palabras reservadas): 15 DIV 5 = 3 y 15 MOD 5 = 0; 31 DIV 7=4 y 31 MOD 7=3. Tenga cuidado si es posible que sus datos sean negativos, dado que la división entre enteros no está definida para valores negativos del denominador. Con ambos tipos de división, de reales y de enteros, todo intento de división por cero resultará en un error en tiempo de ejecución: al menos hasta que alguien invente alguna forma de calcular el infinito.

En una expresión, por supuesto, las operaciones de multiplicación y división se evalúan antes que la suma y la resta. Debe utilizarse la notación habitual de paréntesis para evitar esta "precedencia" lógica. Por ejemplo: (8+4)DIV 2=6, pero 8+4 DIV 2=10.

Funciones de transferencia

Aunque no podemos efectuar una asignación directa de un valor real a un entero, el PASCAL proporciona "funciones de transferencia" muy útiles, con los identificadores trunc y round, que se pueden emplear para llevar a cabo esta asignación. Trunc simplemente trunca un número real, independientemente de su parte fraccional, de modo que trunc (3.999)=3 y trunc (−123.456)=−123. La función round realiza un redondeo inteligente, a cero si la parte fraccional es inferior a 0.5 y alejándose de cero, en caso contrario: por consiguiente, round (1234.5)=1235 y round (−0.49237)=0. Naturalmente, se producirá un error si el argumento real de



alguna de estas funciones diera un resultado entero que cayera fuera de la escala del "tipo" *integer* (entero) (de −MaxInt a MaxInt), de modo que compruébelo siempre primero con una sentencia IF. Asimismo, el argumento *debe* ser real, no entero (éstos, al fin y al cabo, ya están truncados y redondeados).

Hay un bonito detalle que no les resultará familiar a los programadores que han desfallecido ante expresiones que utilicen la función INT del BASIC. El PASCAL posee una función predefinida que devuelve un resultado booleano: *false* si su argumento entero es un número par y *true* si es impar. El identificador utilizado se denomina odd.

```
IF odd(N)
  THEN
    WriteLn('Es impar!')
  ELSE
    WriteLn('Es par!')
```

o:

```
IF odd(N)THEN
  IF N MOD 2=0 THEN
    WriteLn ('Atrapa al compilador!')
```

La tabla muestra todas las funciones aritméticas del PASCAL. Todas pueden tomar cualquier argumento numérico simple, real o entero, y las dos primeras, abs y sqr, devolverán un valor de tipo compatible con su argumento. Por consiguiente: abs (−19.372) es 19.372, y abs(255) es 255. Las otras funciones, sin embargo, *siempre* darán un resultado real, de modo que sqrt(16) es 4.0, no 4. Ello se debe a que los algoritmos utilizados para evaluarlas están basados en el sumatorio de una serie de términos, todos los cuales son fraccionales.

Aquí encontramos una importante lección de estilo con respecto a la comparación de reales. Observe que al referirnos a los resultados reales de las funciones aritméticas hemos utilizado la palabra "es" en lugar de un signo de igualdad. Lo que queremos expresar es "es lo mismo que" y no "es igual a", porque *nunca* es seguro comparar valores reales a un grado exacto de igualdad. El más mínimo error de cálculo significará que dos reales nominalmente "iguales" puedan en realidad diferir en, supongamos, 1.0E-27. Es insignificante, por supuesto, pero el ordenador no lo sabe. En vez de utilizar IF X=Y THEN..., podemos utilizar la función predefinida abs del PASCAL:

```
IF abs(X-Y) < Insignificante THEN {etc.}
```

Aquí comprobamos el caso en el cual la diferencia entre X e Y es sumamente pequeña. Sería muy útil especificar en una definición CONST al comienzo del programa exactamente qué consideramos insignificante. Los logaritmos se dan en su base natural (e), no 10, y exp eleva e a la potencia de su argumento (es decir, se "exponencia"). El PASCAL no posee un verdadero operador de exponenciación, lo que obedeció a una decisión de diseño que tomó Wirth de forma deliberada. En muchas ocasiones usted habrá visto en programas en BASIC cosas tan tontas como:

```
500 LET D=B^2+4*A*C
```

que utiliza la forma más lenta y menos exacta posible de calcular el cuadrado de un número. B*B es muchísimo mejor, por supuesto. Observe que sqr(N) del PASCAL devuelve el cuadrado de N direc-

La sección dorada

Existe un solo punto en la línea que la divida en dos partes de modo tal que la proporción de la longitud de la sección más pequeña respecto a la grande sea la misma que la proporción de la mayor respecto a la longitud de la línea. Esta proporción se denomina *sectio aurea* o *sección dorada* y es un número irracional con una cantidad infinita de dígitos

La sección dorada la utilizaban los artistas y arquitectos griegos (especialmente para el diseño del Partenón) y suele revivirse en las teorías de proporción. Sin embargo, algunas de sus manifestaciones más familiares se encuentran en la naturaleza. La espiral de "concha marina" se basa en ella, dado que sus vectores de radio, separados por 90°, se hallan en esta proporción de secciones

Dentro del rectángulo ABCD se puede trazar un cuadrado ABEF, de modo que se forme un rectángulo (EFDC) con la misma proporción de lados que ABCD. El proceso se puede continuar *ad infinitum* y los vértices de los cuadrados son puntos situados en la "espiral de sección dorada"

tamente como un entero verdadero (asumiendo que N es un entero); sqr(X/3) daría un real. Observe, asimismo, otro hábito del BASIC que habrá de desterrar: sqrt(it), *no* sqr(it) si desea la raíz cuadrada de un número.

Tipos de subrango

Todos los tipos escalares poseen un rango de valores definido y ordenado, y se los puede utilizar como un tipo "anfitrión" del cual se pueden derivar nuevos tipos de subrangos. Éstos se componen, como sugiere su nombre, de una porción contigua del tipo anfitrión. Los límites inferior y superior necesarios para delimitar el subrango se listan entre el signo de "elipsis" (..) en la definición del identificador de tipo de subrango; por ejemplo:

```
TYPE
  byte=0..255 {subrango de integer}
  alpha='A'..'z';{subrango de char}
  baraja=(trebol, diamante, corazon, pica);{un tipo
        nuevo}
  mayor=corazon..pica; {subrango de baraja}
```

Aparte de las ventajas que hemos analizado antes, un buen compilador optimizador de PASCAL almacenará variables de tipo byte, por ejemplo, en ocho bits en vez de los 32 posibles (si se emplea la representación de entero de cuatro bytes). Por consiguiente, una gran matriz de este tipo ocupará sólo el 25 % de la memoria que se utilizaría con elementos enteros no cualificados. Observe que el tipo mayor tiene dos valores posibles, algo así como pensar en la definición de valores booleanos como:

```
TYPE
  booleano=(false, true);
```

Todos los tipos de subrango heredan de su tipo anfitrión tanto la clasificación de datos como las operaciones en él definidas.

La definición anterior de *alpha* podría permitir, por supuesto, otras cosas además de letras del alfabeto. En particular, el juego de caracteres ASCII posee otros seis símbolos en el espacio entre "Z" y "a", incluyendo los símbolos de los corchetes ('['y']'). Éstos también figuran entre los 23 símbolos reservados del PASCAL, y se utilizan para delimitar los componentes de algunos tipos de datos estructurados, como en muchos lenguajes. Originalmente el BASIC también empleaba estos símbolos para encerrar los índices de matriz, pero la sintaxis se cambió por los paréntesis debido a que los ordenadores antiguos a veces poseían juegos de caracteres "medio ASCII" que no incluían letras en minúscula, corchetes, las llaves que utiliza el PASCAL para encerrar los comentarios, y algunos otros símbolos. El Apple II sin modificar también adolecía de esta restricción.

El lenguaje PASCAL proporciona las siguientes alternativas opcionales para los casos en que exista este problema: se permite (.y.) en lugar de [y], y (*y*) se pueden utilizar en lugar de {y}. La segunda alternativa, para delimitadores de comentarios, es bastante común; no obstante, es probable que sólo los compiladores autorizados por la ISO soporten las alternativas de los corchetes. Se puede realizar cualquiera de las sustituciones o ambas, de modo que: (*este es un comentario legal}.

Función	Valor devuelto
abs (K)	valor absoluto de K
sqr (K)	cuadrado de K
sqrt (K)	raíz cuadrada de K
sin (A)	seno de A radianes
cos (A)	coseno de A radianes
arctan (T)	ángulo (en radianes) con tangente T
ln (K)	logaritmo natural de K
exp (L)	*e* elevado a la potencia L ("anti-Ln" de L)

Banda de oro

El programa Dorada, que ofrecemos aquí, genera términos Fibonacci y los imprime junto con su proporción. Utilizando el programa, se puede ver que una de las propiedades interesantes de la serie de Fibonacci (1,1,2,3,5,8,13, etc.) es que la proporción de cada par sucesivo de términos se aproxima cada vez más al valor de la sección dorada. El programa demuestra algunos de los principios que ya hemos analizado en el curso, incluyendo nuevos ejemplos de la estructura REPEAT del PASCAL. A modo de ejercicio, trate de alterar la condición para la finalización del bucle de modo que se detenga cuando el siguiente término Fibonacci a calcular exceda el valor MaxInt

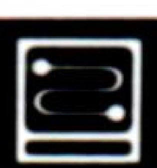
```
PROGRAM      Dorada         (output);
CONST
         Epsilon        =1.0E-08;
TYPE
         Fibonacci      =1..MaxInt;
VAR
         primero,
         segundo,
         siguiente   :Fibonacci;
         proporción,
         Oro         :real;
         contador    :integer;
BEGIN
 WriteLn ('La seccion dorada' : 30);
 WriteLn;
 WriteLn ('Series Fibonacci :');
 primero := 1;    {por definicion}
 segundo :=1;
 proporcion := primero/segundo;
 contador :=0;

 REPEAT
   IF contador MOD 10 = 0 THEN
     BEGIN {Cabecera cada 10 lineas}
       WriteLn;
       WriteLn ('Primero' : 10,
         'Segundo' : 10, 'Proporcion' : 14);
       WriteLn
     END;

   WriteLn (primero : 10, segundo :10,
         proporcion : 16 : 8 );
   contador := contador+1;
   Oro := proporcion;  {recordar antiguo GS}
   siguiente := primero + segundo;
   primero := segundo;       {ir avanzando hacia ..}
   segundo := siguiente;  {arriba de la serie}
   proporcion :=primero / segundo

UNTIL abs (proporcion - Oro ) < Epsilon;

WriteLn;
WriteLn ('El termino medio dorado es : ',
      100*proporcion : 10 : 5, '%' )
END.
```

"Interrumpimos este programa para..."

Con señales denominadas "interrupciones" y "eventos" el OS gestiona los recursos de proceso de un ordenador

Cuando se ejecuta un programa en BASIC con un BBC Micro, es fácil creer que todas las energías de la máquina están concentradas en esa ejecución. Pero no es así; constantemente se están realizando otras tareas como la lectura del teclado por parte del OS. Este aparente "tiempo compartido" de las potencialidades del ordenador es posible gracias al empleo de las *interrupciones*. Una interrupción es, en esencia, una señal dirigida a la CPU que la instruye para que detenga lo que esté haciendo en ese momento y realice otra tarea. Una vez realizada esta última tarea, la CPU vuelve a su trabajo original y continúa con él como si nada hubiera ocurrido. Las interrupciones en el BBC Micro se encargan de actividades tan dispares como la lectura del teclado, el parpadeo de colores, el proceso de la instrucción ENVELOPE y las lecturas del convertidor A/D. Ahondemos un poco más en cómo realiza esto la máquina.

Las interrupciones en el BBC Micro se generan mediante un buen número de dispositivos hardware incorporados al sistema del ordenador; entre ellos se encuentra el chip de Interface Serial, los VIA del sistema y del usuario (VIA: adaptador de interface versátil), un equipo físico Econet e Interface de Discos. Estos dispositivos originan dos tipos de interrupciones en el procesador 6502. Ambos *causan* que el sistema interrumpa lo que está haciendo, aunque se puede instruir a la CPU para que ignore uno de los dos tipos si se desea. Ésas son llamadas *peticiones de interrupción enmascarables,* o bien IRQ (*maskable interrupt requests*). El otro tipo de interrupción no será ignorado nunca por la CPU, convirtiéndose en una interrupción absolutamente prioritaria. Este tipo es la llamada *interrupción no enmascarable,* o bien NMI (*non-maskable interrupt*).

Las NMI en el BBC se reservan para dispositivos que necesitan una atención inmediata de la CPU, mientras que las IRQ son generadas por los dispositivos que solicitan la atención de la CPU, pero pueden esperarse un poco. Los únicos dos dispositivos que usan las NMI son el hardware de la red Econet y el sistema de ficheros en disco. Al recibir una NMI, el OS emplea una pequeña rutina en código máquina situada en la página &0D. Ésta es una de las razones por las que la página &0D queda a libre disposición del usuario una vez ajustada la interface para discos. A causa de estas propiedades especiales de la NMI, no la vamos a considerar aquí. Al programador se le desaconseja su uso en concreto por la misma Acorn.

Nos quedamos, pues, con las IRQ. Las interrupciones enmascarables del procesador 6502 pueden quedar desactivadas por medio de la instrucción SEI desde programas de ensamblador; y reactivadas por medio de la instrucción CLI. Cuando es recibida una IRQ por el 6502, se actúa según la fuente de la interrupción. El 6502 descubre el origen de la interrupción "preguntando" a cada posible fuente de interrupción si fue la causa de la interrupción. Tan pronto como se ha encontrado cuál es la fuente, la interrupción es manejable y el dispositivo que la generó es instruido para que olvide la interrupción hasta que ocurra otro evento que merezca el envío de una nueva interrupción al 6502. A esto se le llama *borrar* la interrupción. La rutina en código máquina que maneja la interrupción se denomina ISR (*interrupt service rutine:* rutina de servicio de interrupciones).

¿Qué hace una ISR?

Dado que el programa interrumpido no debe acusar cambio alguno en los valores existentes en los registros del 6502, la primera misión de una ISR es guardar los valores de los registros. Esto se hace en la pila del 6502 por medio de una serie de operaciones de carga (*push*). Así, una vez tratada la interrupción, estos valores pueden ser restaurados. La ISR debe posteriormente realizar todo lo necesario para servir la interrupción, borrar la condición de interrupción antes de abandonar la ISR y devolver el control al programa interrumpido. La razón por la que es tan importante que se borre la interrupción es obvia; si no se borrara, tan pronto como se devolviera el control al programa interrumpido el 6502 volvería a recibir una IRQ nuevamente desde la fuente, poniendo la CPU en un bucle sin salida.

Una vez sentados los principios generales del tratamiento de una interrupción, veamos el caso específico de un BBC Micro. Este ordenador se apoya en gran medida en las interrupciones, sirviéndose de ellas para un gran número de operaciones. Por esta razón, hay que tener cuidado de no desactivar las interrupciones demasiado tiempo por medio de SEI. Si cuidamos que el intervalo de tiempo de desactivación de las interrupciones sea inferior a unos pocos milisegundos, no ocurrirá nada especial. Pero un poco más de tiempo provocará un extraño comportamiento del OS. El siguiente programa demuestra lo que sucede en un BBC Micro cuando se desactivan las interrupciones durante diversos períodos de tiempo:

```
10 MODE 2
20 DIM C (100): REM establece el espacio para el
   cod. maq.
30 FOR I%=0 TO 2 STEP 2
40 P%=C
```

```
50 [OPT I%
60 .code SEI/desactiva las interrupciones
70 RTS
80 ]:NEXT
90 COLOUR 14
100 PRINT "Hola!"
110 ENVELOPE 1,1,4,-4,3,10,20,20,127,0,0,
    -5,126,126
120 SOUND 1,1,160,200
130 TIME=0
140 REPEAT
150 PRINT TIME,I%
160 I%=I%+1
170 IF TIME > 100 THEN CALL code
180 UNTIL FALSE: REM no acaba jamas
```

Ejecute este programa y verá que en cuanto se desactivan las interrupciones llamando a la rutina en código máquina denominada code, la variable TIME no se altera y se detienen tanto los colores parpadeantes como el proceso ENVELOPE. Pero la máquina no se "cuelga", ya que la rutina continúa imprimiendo I% fielmente. Asegúrese de pulsar CTRL BREAK cuando haya finalizado la ejecución de este programa.

El vector IRQV1

Cuando el 6502 recibe una IRQ, el control se pasa a una ISR cuya dirección está contenida en un vector llamado ISQV1. Este vector está en la dirección &204 y &205. La rutina entrada por medio de este vector trata las interrupciones a través de tres dispositivos hardware: el chip de Interface Serial, el VIA del sistema y parte del VIA del usuario. Toman el orden de prioridad en que se han citado.

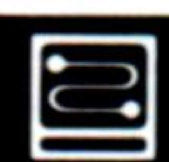
El chip de Interface Serial es una pieza del hardware que se encarga de la interface RS423 y de la interface de cintas. Las interrupciones llegadas por esta fuente conciernen, claro está, a estas interfaces. El chip es el dispositivo de máxima prioridad en el uso del vector.

Por este orden tenemos a continuación las interrupciones generadas por medio del VIA del sistema. Este dispositivo de tanta utilidad interrelaciona la CPU con otros muchos dispositivos de la máquina, y ofrece además algunas facilidades de temporizado. Una interrupción a través del VIA del sistema se obtiene al pulsar una tecla, que indica al OS que debe procesar la tecla de algún modo. Se produce también una interrupción por este chip cuando se envía un fragmento entero de información a la pantalla o al televisor. Tal interrupción se envía una vez cada 1/50 de segundo, y se conoce como *interrupción de sincronía vertical*.

El convertidor A/D actúa también en conjunción con el VIA del sistema, enviándonos una interrupción cada vez que se finaliza la conversión de una señal. Esta interrupción particular hace que el OS actualice los valores empleados para devolver un número a la función ADVAL. Otra última interrupción importante proporcionada por el VIA del sistema se genera una vez cada centésima de segundo. Hace que se incremente la variable TIME, que se actualice el "reloj de intervalos" (empleado para generar eventos), que se decremente el temporizador INKEY (que cronometra la demora en una instrucción INKEY) y que se procese parte de un ENVELOPE. También provoca que el OS continúe con el proceso de cualquier tecla pulsada. Otras interrupciones proporcionadas por medio del VIA del siste-

¡Camarero!
No es difícil imaginar la CPU del BBC Micro desempeñando un papel semejante al de un camarero en un restaurante muy concurrido. Lo mismo que el 6502 debe repartir su atención entre los periféricos, la transferencia de datos y la ejecución de programas, así un camarero debe atender a los clientes importantes (que solicitan preferencia sobre otros), limpiar las mesas y solucionar eventuales situaciones conflictivas.
Un camarero eficaz debe ser capaz de repartir su atención a cada una de estas tareas cuando se soliciten y además volver posteriormente a su punto de partida, la tarea menos prioritaria que dejó, de tal modo que el servicio de la cocina al comensal sea fluido. El 6502 obra del mismo modo, atendiendo a las "interrupciones" y a los "eventos" que el hardware o el control del software han generado. Una vez servida la interrupción, la CPU puede volver a su tarea anterior

Kevin Jones



ma se refieren al sistema sintetizador de la voz y al lápiz fotoeléctrico si se dispone de él. Existen las llamadas OSBYTE que nos permiten alterar la forma en que responde el OS a las interrupciones desde estas dos fuentes. Pero, como pronto veremos, no es aconsejable mezclarlas con el IRQV1.

La interrupción de menor prioridad tratada por este vector se pasa a una Rutina de Servicio de Interrupciones cuya entrada se produce mediante el VIA del usuario, programable por el usuario para obtener un buen abanico de interrupciones en la CPU. Normalmente tales interrupciones serán ignoradas por ésta.

El vector IRQV2

El vector IRQV2, contenido en las posiciones &206 y &207, puede ser alterado por el usuario para que apunte a una ISR creada por él. Hay que usar siempre este vector si deseamos añadir nuestras propias Rutinas de Servicio de Interrupciones, aunque es posible alterar el contenido de IRQV1 para apuntar a una nueva rutina. Este proceso de cambio del contenido de un vector que le obliga a apuntar a una nueva rutina es llamada *intercepción* del vector. Si se desea interceptar el vector IRQV2 para añadir rutinas de interrupción propias, se deben cumplir las siguientes condiciones a la entrada de la rutina apuntada por el vector:

- La CPU debe tener ya almacenado el contenido del registro A en la posición &FC y haber colocado el registro indicador de estado del procesador (PSR) en la pila.
- Los registros X e Y deben estar en el estado que tenían cuando ocurrió la interrupción.

El segundo punto quiere decir que se deben guardar los registros X e Y a la entrada de la propia ISR empleando las siguientes sentencias:

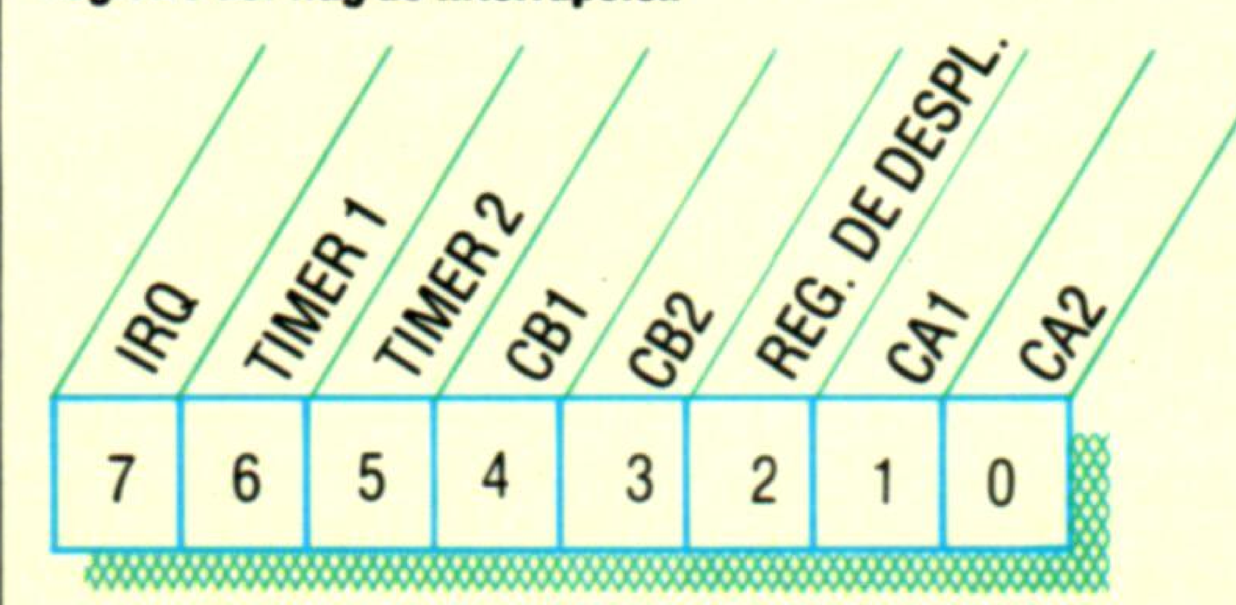
```
TXA:PHA
TYA:PHA
```

La rutina debe borrar la condición de interrupción. Esto depende de la fuente de la interrupción: si se trata del VIA del usuario, por lo general implica la lectura y escritura de ciertos registros del VIA del usuario. Antes de volver al programa interrumpido, hay que recordar restablecer los registros y emplear la instrucción RTI para abandonar la rutina de interrupción. Y finalmente se advertirá que aunque el registro A se almacena en la posición &FC al entrar a IRQV1, una interrupción ulterior puede causar su sobreescritura antes de acabar de procesar nuestra interrupción del usuario. Dos soluciones pueden darse: o bien ejecutar una instrucción SEI a la entrada de la rutina propia y una CLI antes de abandonarla, o bien llevar el registro A a la pila en la entrada de dicha rutina y restaurarlo antes de abandonarla. Si se hace esto último, debe también almacenar el valor del registro A en &FC antes de ejecutar la instrucción RTI. Los usuarios con poca experiencia del VIA del usuario harían bien en tomar todas las precauciones.

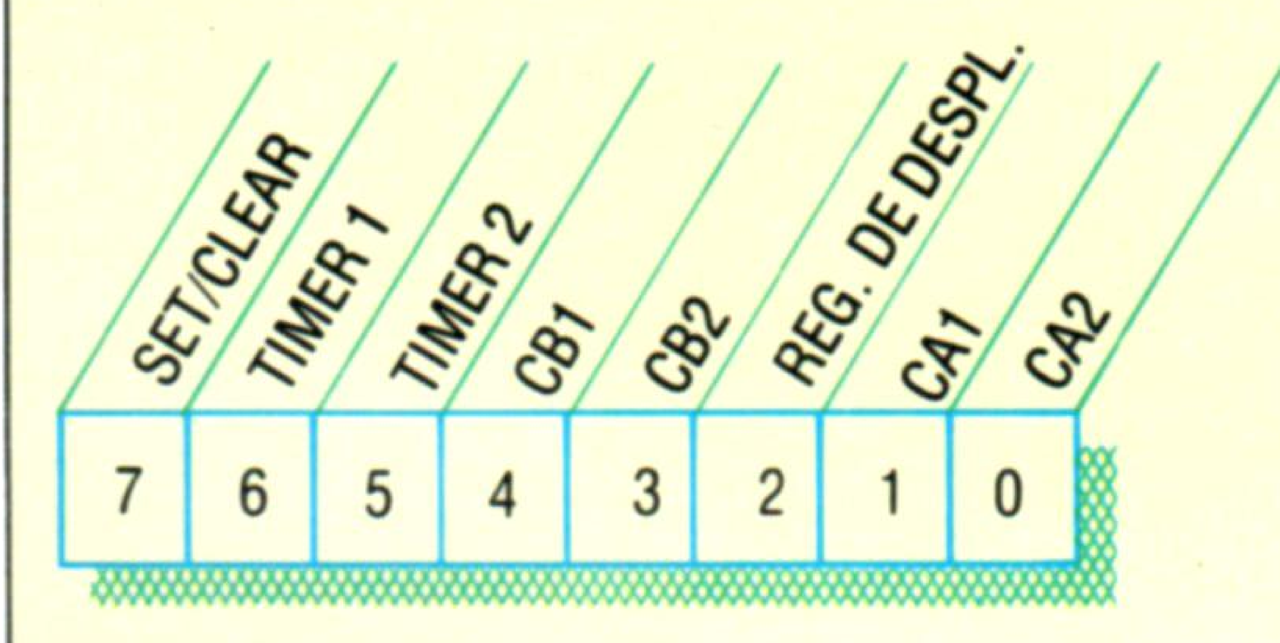
El BBC Micro emplea señales de "evento" como alternativa al uso de las interrupciones. Estas señales (que algunos consideran "interrupciones sin lágrimas") constituyen el tema que desarrollaremos en nuestro próximo capítulo.

Interrupciones del chip VIA

Registro del flag de interrupción



Registro de activación de la interrupción



Liz Dixon

Aunque el procesador 6502 del BBC MIcro tiene sólo dos líneas de interrupción (la IRQ y la NMI), los chips VIA de interface con periféricos pueden ser programados de tal modo que muchos dispositivos puedan compartir la línea de interrupción IRQ para el procesador. Cada chip VIA tiene dos puertas de E/S, cada una de las cuales tiene ocho líneas de datos y un par de líneas de señal. En la puerta A, son las denominadas CA1 y CA2, y en la puerta B, son CB1 y CB2. Estas líneas son a menudo empleadas para el apretón de manos, es decir, para la transferencia de datos sincronizada entre el micro y un dispositivo periférico, pero pueden también ser programadas para que actúen como líneas de interrupción. Además de las cuatro posibles líneas de interrupción que entran en cada VIA hay una interrupción interna que puede ser generada por el registro de desplazamiento del chip. Este registro se usa para la transmisión en serie por medio del CB2 en respuesta al impulso de una fuente externa sobre CB1. Finalmente hay también dos relojes de intervalos de 16 bits que pueden también generar interrupciones. Todas ellas deben compartir una única línea IRQ hacia el procesador. Cuando una interrupción se genera, el procesador puede que desee conocer la fuente de la interrupción. Esto es posible interrogando el registro del flag de interrupción en el VIA. Cada bit de este registro corresponde a una de las posibles fuentes y se pone a uno al solicitar la fuente una interrupción. El bit 7 es una excepción, pues se pone a uno si cualquier dispositivo pide una interrupción. Y es que el bit 7 es una línea IRQ. Pueden existir ocasiones en que deseemos detener un dispositivo en su interrupción mientras el procesador sirve otra diferente. Lo podemos conseguir programando el IER (registro de activación de la interrupción). Éste tiene ocho bits que corresponden a los mismos dispositivos que los bits del anterior IFR (registro del flag de interrupción). Si se pone un bit a uno se reconocerán las interrupciones del correspondiente dispositivo; si el bit está a cero éstas son ignoradas

Fortaleza espacial

"Zaxxon" es un programa con excelentes gráficos tridimensionales, creado para las salas recreativas, que ahora está disponible para micros personales

La última frontera
Aquí vemos tres momentos de *Zaxxon*. El jugador inicia el juego pilotando una nave espacial hacia la primera de las dos fortalezas. La nave se debe maniobrar en el ángulo y la altitud correctas para poder atravesar la pared. En el interior de la fortaleza propiamente dicha hay otra pared, que está protegida por un escudo eléctrico. Se pueden conseguir puntos extras destruyendo las torretas con cañones. Una vez salvada la primera fortaleza, la nave sale al espacio, donde debe disparar contra gran número de naves enemigas que pasan junto a ella. Por último, tras superar las defensas de la segunda fortaleza, la nave se enfrenta directamente con el robot Zaxxon

Ian McKinnell

El objetivo de *Zaxxon* es pilotar una nave espacial sorteando numerosos obstáculos en un intento por destruir al robot Zaxxon. Al principio su nave se halla en la inmensidad del espacio; finalmente aparece delante de usted una "fortaleza espacial", rodeada por una alta muralla. Para poder seguir adelante, debe pasar a través de un agujero que hay en esta barrera, antes de enfrentarse al primero de los obstáculos que encontrará en su camino. Éstos incluyen misiles, pantallas de radar y depósitos de combustible, todos en realistas gráficos tridimensionales.

Los depósitos de combustible son sumamente importantes puesto que su destrucción supone un aumento del combustible de su nave espacial.

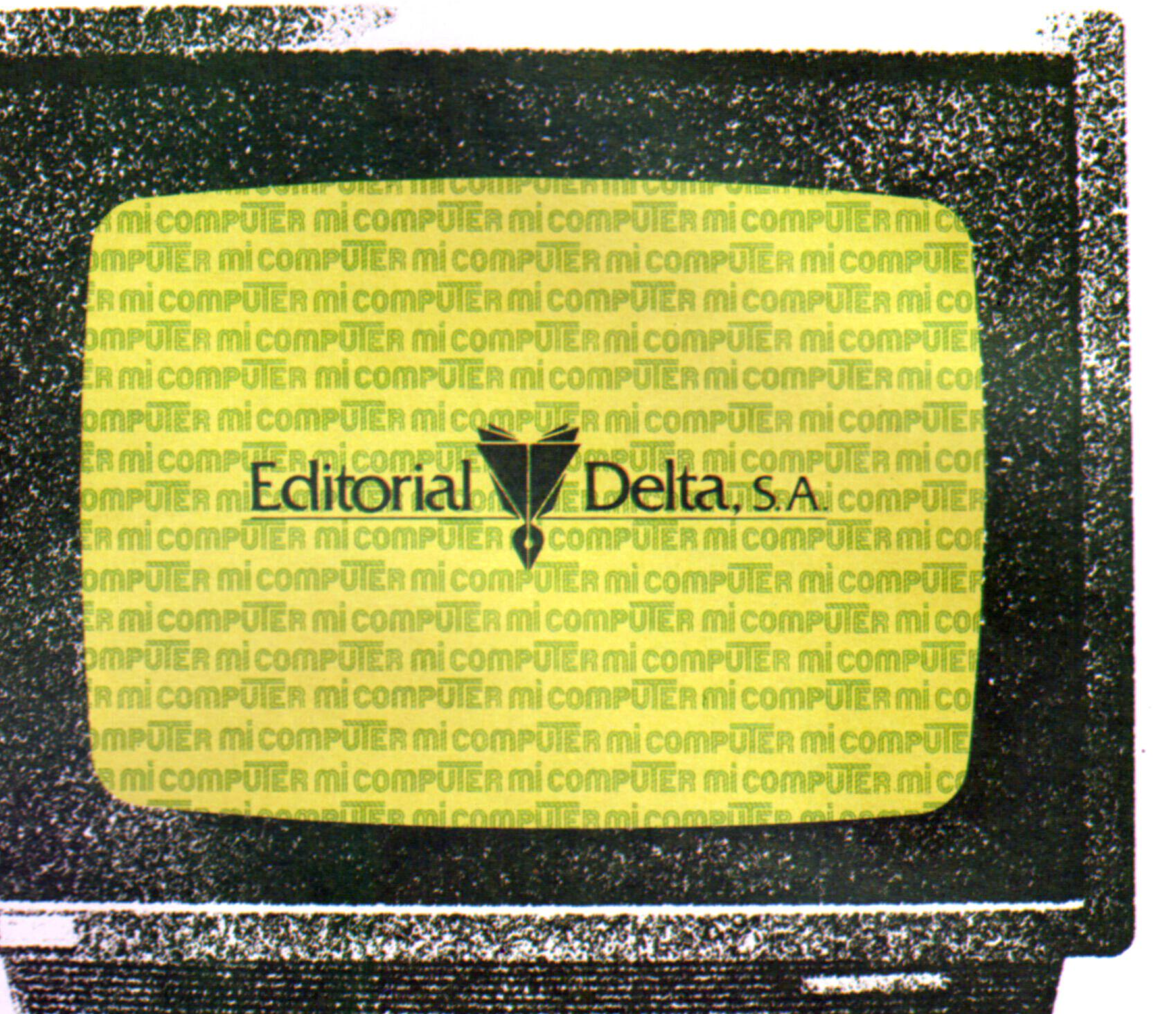
El diseño de los gráficos es de gran belleza. Se utilizan tres sprites diferentes para representar la nave del jugador: uno para los picados, otro para los ascensos y el tercero para el vuelo normal. A medida que la nave se eleva o desciende en picado, el tamaño de estos sprites aumenta y disminuye de una forma muy convincente. Los obstáculos de tierra, asimismo, están bien dibujados y la perspectiva le otorga cierto nivel de realismo al juego. Un detalle especialmente atractivo es la sombra de la nave espacial; ésta va creciendo a medida que la nave pierde altura y pasa a través de los objetos que se hallan en tierra.

Si durante esta fase del juego su nave es destruida por el fuego enemigo, usted debe regresar al comienzo, pero al adquirir cierta práctica descubrirá que es posible sortear la mayoría de los peligros. El esfuerzo que supone destruir las pantallas de radar vale la pena, porque éstas, de dejarse solas, hacen que los misiles de desplazamiento horizontal se dirijan hacia usted. Antes de poder destruir un radar, se requieren tres impactos directos. En todo momento se debe tener un gran cuidado en conducir la nave a la altura correcta: es muy fácil leer de forma errónea el altímetro y volar directamente hacia un misil que se acerque. Finalmente el jugador habrá de enfrentarse con un campo de fuerza eléctrica: debe ascender muy alto para superarlo y luego descender en picado hasta un nivel más bajo, donde podrá destruir los cazas enemigos que lo esperan en tierra. Es aconsejable librarse de la mayor cantidad posible de éstos, ya que todo adversario eliminado en esta etapa desaparece definitivamente del juego.

Después de sortear con éxito las defensas de la fortaleza, su nave vuelve a encaminarse hacia el espacio, donde se enfrenta a un alto número de naves enemigas. En este punto, el usuario se enfrenta con el problema de jugar a un juego tridimensional en una pantalla bidimensional. Para atacar las naves enemigas es necesario ascender o descender hasta el nivel correcto; esto supone ensayo y error, dado que la única indicación que se ofrece son las dimensiones relativas de las diversas naves: la posición de disparo adecuada se produce al adquirir su nave aproximadamente el mismo tamaño que las naves enemigas. Durante esta fase del juego, ¡un movimiento rápido del dedo disparador es una clara ventaja!

Al retirarse otra vez del espacio, su nave regresa a la fortaleza espacial. Esta vez el agujero en la pared es más pequeño y hacer blanco es más difícil. Entonces se encontrará frente a frente con el robot Zaxxon, que avanza hacia usted a través de una rejilla. En este momento, su nave se detiene por completo y su única defensa es vencer al robot antes de que éste lo coja. Ello requiere tres blancos directos sobre los lanzadores de misiles del autómata. Mientras intenta denodadamente conseguirlos, el robot Zaxxon avanza de forma inexorable: si éste consigue alcanzarlo, usted habrá de regresar al comienzo del juego.

Zaxxon: Para el Spectrum y el Commodore 64
Editado por: US Gold, Unit 10, Parkway Industrial Estate, Hineage St, Birmingham B7 4LY, Gran Bretaña
Autor: Peter Adams
Palanca de mando: Necesaria
Formato: Cassette (Spectrum, Commodore); disco (sólo Commodore)


Editorial Delta, S.A.
mi computer



10078

9 788485 822836